REFERENCE MANUAL

SCOP++ 5.5

VIENNA
UNIVERSITY OF
TECHNOLOGY

INSTITUTE OF
PHOTOGRAMMETRY
AND REMOTE SENSING

Trimble

SCOP++ Manual for Version 5.5 and higher

Contents

Part I.

Overview

1. Purpose

SCOP++ is designed for interpolation, management, application and visualization of digital terrain data, with special emphasis on accuracy. SCOP++ aims at high quality of interpolation and of all products derived of the model, and for processing huge amounts of DTM data.

SCOP has been developed and continuously improved over the last 30 years in cooperation of Trimble Germany, Stuttgart, and of Institute of Photogrammetry and Remote Sensing (I.P.F.), Vienna.

SCOP++ is an object oriented (C++), integrated program system with a central graphical user interface including the main graphics panel. For users, a major advancement is the integrated logic to handle user interactions and to supervise processing.

Data and other information management is performed by a specific internal database system (TOPDB) integrated with SCOP++.

Computations are performed in the background by algorithmic servers. These are in most cases the modules of `SCOP V3.5`. In the list of functionality above, those `SCOP V3.5` modules not yet integrated under SCOP++ are marked by a footnote. These concern in the first line *Scop Perspectives*.

In integrating further functionalitity of `SCOP V3.5` into SCOP++, preference will be given to functions explicitly demanded by users.

cmL/cmF (standing for commandLine/commandFile) represent a further stage in the DRE - DRE-X line of controlling processes. cmL allows for controlling the GUI (Graphical User Interface) by commandLine entries - commandLine being a special one-line window to appear on the screen on user request. Commands under cmL consist of words identical with different labels on the screen (of buttons and other GUI elements), and of different symbols to control action. Command files cmF contain user-written procedures consisting of cmL statements, and of cmF-specific commands.

Functionality of SCOP++ is subdivided into a Kernel, and into several packages of extended functionality: Analyzer, Visualizer, LIDAR, TopDM.

1.1. SCOP Kernel

- Input/Output:
  - Data: SCOP Winput (ASCII or binary), AutoCAD DXF, XYZ (ASCII or binary), LAS, and ArcInfo Generate (conversion allowed)
  - Images:
    - raster: SCOP PIX, GeoTIFF (including tiled TIFF and TIFF/JPEG), JPEG

    - vector: SCOP ZWIFI, HPGL, DXF
  - DTM (import): SCOP RDH, ArcInfo ASCII Grid, Raw Binary, TIFF, USGS DEM, USGS SDTS, SRTM
  - DTM (export): SCOP RDH, ArcInfo ASCII Grid, Raw Binary, VRML, AutoCAD DXF, SCOP Winput (ASCII or binary), XYZ (ASCII or binary), STL, DTED, DGM-Band, XYZ-Slope
  - Combined vector/raster: output as PDF
- DTM interpolation with or without filtering (using linear prediction)
- Overlaying DTMs (eg elevations, soil quality, slope, different epochs, and the similar)
- Integrated raster and vector graphics
- Overlaying transparent raster graphics (e. g. a hill-shading over a digital map)
- Editing data supported by the integrated graphics, with automatic updates of the DTM
- Derivation of contour lines (isolines) with cartographic quality
- Basic Profiling
- Z-coding (color-coded height views)
- Hill-shading
- Z-interpolation (e. g. for supplied 2D data)
- Basic image processing

1.2. SCOP Analyzer

- Profiling including cross sectioning
- DTM Algebra (including the derivation of difference models)
- DTM Classification (eg of a difference model; this yields volume computation)
- Derivation of terrain slope and exposition

1.3. SCOP Visualizer

- Perspective views; Panoramic views; Silhouettes
- Annotation of geographic names
- Visibility models and maps[1]
- Extended image processing
- Map of slope vectors
- Skyplots: horizon for a specified point on or above the terrain[1]
- Mono-plotting (Data acquisition from single images by means of a DTM)[1]

1.4. SCOP LIDAR

- Robust Filtering for DTM interpolation
- Blunder detection in input data via DTM interpolation by the methods moving average or moving tilted plane
- Affine filtering
- Specific methods for processing data from laser scanning

1.5. SCOP TopDM

SCOP++ closely co-operates with the *Topographic Data Management* system *TopDM*: a database management system designed for storing, managing and archiving country-wide digital elevation information: terrain models, primary data, and additional information. This package enables full database functionality.

[1]This module is not yet integrated under SCOP++. Currently, a stand-alone server is delivered.

2. About this Version SCOP++ V5.5

The latest changes to version 5.5 concern mainly:

- bug fix: synchronization problems with multiple scop++ clients accessing the RPC_tdmServer resolved,
- bug fix: issue with import of shape files with large coordinates resolved.
- change: when interpolating a DTM for a small part of a big area only the relevant part of the data is read into memory,
- bug fix: erroneous message „File ... <overlay>_ssc1.out_ not found.“ with batch mode fixed,
- bug fix: problem with LAS files containing many variable length records fixed,
- bug fix: handling of „snap to grid“ changed to avoid problems when more than one decimal is necessary for grid coordinates,
- bug fix: problem with differing geometries of vector data and image data with perspective views resolved,
- change: resolving slope dependency with sortout/classify steps will only consider grid points for efficiency reasons,
- statistics output to log file for point density,
- interpolation method „moving planes“ is now available with SCOP++ Kernel.
- **SCOP++ is now available as 64bit program** to overcome some limitations of previous versions.

Part II.

Getting Started

3. Some Essentials

3.1. Projects

A *project* is thought to represent some area of interest, covered by none, one or more digital models (*model overlays*) – such as elevation models, slope models, soil quality models, all with or without data, – and similarly by none, one or more digital images (*image overlays*) – such as orthophotos or digitized maps.

SCOP++ can hold *one single project* open for working with it. Nevertheless, multiple instances of SCOP++ can be started for working with multiple projects simultaneously.

3.2. Overlays

Both *model* and *image overlays* are either to be imported to SCOP++, or to be derived by it based on terrain and other data.

For each overlay a corresponding subdirectory will be created by SCOP++. This directory tree will always be created – if for nothing else then just for scratch files, log files and the similar produced by the numerous processes under SCOP++. It is possible to have import and export files at any other place on the disk, and finding/placing them using file browsers; however, it is considerably more convenient to place such files into the corresponding directories of the standard SCOP++ project directory tree.

Standard SCOP++ project directory tree:

```
...\ScopProjects
    \ProjectName1
        \modelOverlay1
        \modelOverlay2
        \...
        \imageOverlay1
        \imageOverlay2
        \...
    \ProjectName2
        \...
    \...
```

3.3. Database

For efficient storage of primary (topographic) vector-type data the internal geo-coded relational database TOPDB is used. This database was developed at the *Institute of Photogrammetry and Remote Sensing* for storing country wide digital elevation data. Although

the SCOP++ user will not get in direct contact with this database, it is important to know that there is one in the background. The database is organized in a way, that for each SCOP++ project one database table is created. This table is physically stored as file with the extension ".TOP" in the projects root directory. When working on the project, primary data is read from files coded in any of the supported data formats and is stored in the project table, being the data pool for the project. One of the major advancements of using a geo-coded database is the possibility of formulating geometric as well as non geometric conditions. This makes it - for instance - easy to divide the whole project area into smaller parts and interpolate these tiles subsequently. Another advantage is, that the editing tools implemented in SCOP++ directly access the database. For that reason changes in the data source are immediately reflected in the calculated surface model and the derived products like isolines, etc.

The project table is in binary format and must not be edited or deleted by the user!! An interface for an Oracle Database is currently under development.

3.4. Abnormal Program Termination

If - for any reason - the program comes to crash (because of bugs in the program, etc.) use the following steps.

- check the Task Manager's list of running processes, and terminate those relevant to SCOP++. This is in the first line `scop++.exe` itself, and also `RPC_TDMserver.exe`. Furthermore, this concerns also the `SCOP V3.5` servers

  `dmssrv.exe`,
  `dtmsrv.exe`,
  `intsrv.exe`,
  `isosrv.exe`,
  `persrv.exe`,
  `pixsrv.exe`,
  `prosrv.exe`, and
  `slpsrv.exe`.
- if further problems occur: It may happen that SCOP++ will not start again. In that case, we recommend to delete the lock-file of the data base manually. You can find it in the directory of your SCOP++-projects in the directory `topdb\sys` under the name `tdm_lock.dat`. For your convenience there is a batch file named `cleanDB.bat` located in directory `topdb` which can be used to reset the database to a clean state.

4. First Steps

This chapter is a tutorial to provide first easy impressions about working with SCOP++. The reader is thought to have SCOP++ running, and applying it to a sample project provided with the installation for this purpose (Ens). Please go step-by-step, guided by the description to follow.

4.1. The Sample Project Ens

The sample project – Ens – is placed into the directory specified for `ScopProjects` at installing SCOP++. Under the directory `Ens` there is a subdirectory `ens2000` (standing for the digital terrain model (elevation model) of epoch 2000). In this directory `Ens\ens2000` you will find the file `ens2000.wnp` to carry measurements coded according to the SCOP-specific format WINPUT. (There are numerous other input formats provided – see further chapters in this manual). SCOP++ will recognize the file `ens2000.wnp` as data input for this given *model overlay* based on the file name being identical to the name of the overlay (ens2000), and based on its extension *wnp* being standard for files carrying WINPUT-coded data. This allows for importing the file automatically – if not yet on the internal database of SCOP++.

4.2. Start SCOP++

SCOP++ will be started in ways usual under Windows (XP/Vista/Windows 7). You can find it as

```
<InstallPath>\SCOP++\bin\scop++.exe
```

The installation program places a folder with a shortcut to SCOP++ into the Windows-Start-menu and additionaly onto your desktop. You can use one of this shortcuts to start SCOP++.

When started, SCOP++ will display its main window. On the left side of this window there are two labels: *Model Overlays*, and *Image Overlays*. At this point, the list below them is empty.

The following step-by-step guidelines describe how to create a *new* project with the name "Ens". If the project Ens was already opened once, SCOP++ will not allow creating a new project with the same name. Thus, before going on, check whether the project is already registered and – if so – deregister it at the database. This can be done by opening the drop-down menu *Project/Deregister....* A window will appear showing all projects registered at the database. If the project Ens is included in this list, select it and press button OK.

Now, you can use the drop-down menu *Project/New* to create a new project. The dialog window *New Project* will be displayed for specifying the name of the new project and the full path of the project directory. Type the name of the project

```
Ens
```

into text field *Name* and click the button *OK*. The name of the project (Ens) will automatically be added to the path of the SCOP Projects directory specified at installation time. The appropriate directory will be created - if not yet existing - and will be used as root directory for all files of this project. If the project already exists, you are informed that the project file `Ens.spr` will be overwritten.

Creating a new project will also bring up the dialog window *Add model*. Type the name of the *model overlay* (ens2000) into text field *Name*. Make the proper choice of the group *Contents of Overlay*:

- make a model from terrain or other data (radio button *Both* – the default),
- import a DTM-file already on the disk (radio button *Model-only*), or
- add an overlay consisting just of data with the sole purpose to be displayed in conjunction with other model and image overlays (radio button *Data-only*).

In our example, the default *Both* is the proper choice. Clicking at button *OK* will

- place a button *ens2000 ...* under the label *Model Overlays* on the left side of the window,
- the default data file `ens2000.wnp` will be imported into the internal database,
- the window *Limits* will be displayed, and an extract of the contents of `ens2000.wnp` will be displayed on it,
- and, internally, default values for different processes will be derived from the data on `ens2000.wnp`.

Let now exit and re-start SCOP++. You can exit it either by clicking the cross at the right upper corner of the main window, by clicking at the miniature SCOP++ icon in the upper left corner of the main window – and clicking at *Close*, or by clicking at the drop-down menu *Project/Exit*.

Now re-start the program the same way as you started it the first time. Click at the drop-down menu *Project/Recent projects*. Click there at Ens; this will re-open the project, re-build window *Limits*, and place button *ens2000*.

4.3. Model Overlay ens2000

For a fast impression, without thinking a lot: click at the button *ens2000*. A narrow window *ens2000* will open, with a series of buttons on it – corresponding to *methods* to operate on this *model overlay*. Left-click the button *Data*, and then in the pop-up menu, the field *display*. This action will display the break and border lines in the main graphics panel. Now repeat this same action at buttons *Isolines..., Shade..., Z-code...* – i.e. left-click them, and choose the field *display* in the pop-up menus to appear. If the digital surface of

ens2000 is not available, the light of the special button termed as state-switch *Model...* will blink for a short while, indicating that the surface will be interpolated. Then the lights of the state-switches *Isolines..., Shade..., Z-code...* will start to blink so to indicate the corresponding processes running – and after a while the main graphics panel will be updated to carry

- the break and border lines from file `ens2000.wnp`,
- the isolines (in this case: the contour lines) derived from the terrain model as interpolated,
- the hill-shading and the Z-coding as derived from the terrain model – integrated (mixed) into a single raster image.

Left-click state-switch *Shade...*, and set its *state* to *off*; this will remove the hill-shading from the main graphics panel, the image becomes very flat.

Left-click state-switch *Shade...* again, and choose *properties...* from the pop-up menu to appear. A window with parameters appears. Close this window by any of the usual means, and then *right*click this same state-switch . This renders the same parameter window open. Choose any *sun position*, and click at button *OK*; the window will close, and nothing else will happen. Why?!

– Because the *state* of state-switch *Shade...* is *off*: in SCOP++, no computations will be performed unless

- the results are needed for any active display (e.g. of hill-shading),
- the results are prerequisite for some other processes switched active (this is why the surface model has been interpolated automatically),
- or, in special cases, an explicit request on behalf of the user is there (an example of this is provided via the button *Interpolate* in the properties' window to state-switch *Model...*).

This *principle* carries the name *Lazy Processing*.

Set the *state* of state-switch *shade...* to *display* – and the hill-shading will be re-computed according to your choice of *sun position*.

Now right-click the state-switch *Model...*, and change the *grid step* of the terrain model, e.g., to 4 meters. Click at button *OK*; this closes the window, and - because some state-switches of this *model overlay* are in an active state (*display*), the model will be re-interpolated, the active displays will be re-computed and re-displayed. These processes are indicated by the lights of the corresponding state-switches blinking.

4.4. Limits

Let's now turn our attention to specifying *areas of interest*, or – in a short form – to specifying *limits*. In the lower right corner of the main SCOP++ window there is a button *Limits ....* Clicking at it will display – or just bring to the foreground – a window (it appeared automatically on reading in the first data file). The checkboxes *Limits for* are there to allow for choosing the purpose of the *limits* to be specified. Per default, this is

for *views* – such as the ones you have seen in the window *ens2000*: isolines, shading, Z-coding and the similar. A left-click into the graphics panel of the window *Limits* (with releasing the left mouse button), and another left-click at another spot of it will create a yellow rectangle there: the *views limits* (please notice that the colors correspond to the colors of the corresponding labels *views*). With any views' state-switches in an active state, the re-computation of the corresponding graphics contents will be started immediately (blinking lights, step-by-step updated main graphics panel).

Clicking at checkbox *Screen* allows for controlling the zoom window of the main graphics panel together with the views limits (which are still selected). Another way to control zooming in and out is given in the toolbar of the main graphics panel itself; try both of them!

4.5. General MODE

In the upper left part of the main window there is a state-switch *MODE*. Left-clicking it brings up a pop-up menu of three modes: *Final, Screen, and Freeze processing*. The large round light indicates the currently selected mode.

MODE Final (gray light) is to allow for the above: in this MODE, final versions of the different products of SCOP++ will be created – according to the current specifications of the user. This might be of crucial importance in certain cases, such as deriving high resolution raster graphics for external use.

MODE Screen (yellow light, the default) causes graphics files to be derived to correspond to the resolution of the screen. As long as this MODE remains active, specifications such as of pixel size will not be available.

MODE Freeze processing will prevent automatic re-computation of graphics displays on different changes all over the SCOP++ system. This mode is very useful, e.g., when editing data, where re-interpolating the model and re-computing all active displays on every change could become cumbersome. This might in some cases also be true for working with *Limits*.

Try the following: set *MODE* to *freeze*. Digitize new limits for *model* in window *Limits*. Now set *MODE* to *screen*. It is only on this last action that the model and all active displays will be re-computed.

4.6. cmL/cmF

cmL stands for command Language, and *cmF* for command File. *cmL* allows for controlling processes from a special window *cmL*, and *cmF* allows for writing procedures on command files, consisting of *cmL* commands.

For a *very* short hint for using *cmL*, make the main window current (click at it anywhere), and type <Alt>c or <Alt>C. A drop down dialog will open. Press key c and the narrow window *cmL* will appear on the screen. This same can be achieved by clicking at the drop down dialog *cmL* and selecting the entry *cmL*.

Type the following into the window *cmL*:

```
ens2000/iso o, shd o, zco o;
```

ens2000/ will open the window *ens2000* (if not yet open), *o* stands for *off* – i.e. isolines, shading and Z-coding will be set off.

Now type

```
ens2000, model, grid 5, ok; iso d, shd d, zco d;
```

And finally, make the window *SCOP++ – Command Line* current (click at it), and type upper arrows. This will recall the previous lines you typed; type a CR (carriage return) if you have found the needed one:

```
ens2000/iso o, shd o, zco o;
```

For a minimal example of applying *cmF*, type the following into the window *SCOP++ – Command Line*:

```
@ens2000;
```

This will call the *cmF procedure @ens2000;* on file

```
\ScopProjects\Ens\Ens.cmF
```

You can inspect it putting the above file in any editor.

4.7. Image Overlays: the Sample Project Albis

This section will give a brief introduction on how to use image overlays. It will be presented using a project in Switzerland (courtesy Federal Office of Topography, Switzerland, www.swisstopo.ch). Start SCOP++ if it is not yet running, and deregister the project albis if it already exits in the list of registered projects (drop-down menu *Project/Deregister...*). Now, create a new project using the drop-down menu *Project/New*. The dialog window *New Project* will be displayed for specifying the name of the new project and the full path of the project directory. Type the name of the project

```
albis
```

into text field *Name* and click the button *OK* (and accept overwriting the project file `albis.spr` and further messages). The appropriate directory will be created - if not yet existing - and will be used as root directory for all files of this project.

Clicking at button *OK* will also bring up the dialog window *Add model*. Type the name of the *model overlay* (TerrainModel) into text field *Name* and choose the default *Both* in the group *Contents of Overlay*:

Now, an image overlay shall be added. For this project, a digital map is available. Using drop-down menu *Overlays/Add image overlay...* will open the dialog window *Add image*. Type the name of the image overlay in the text field *Name* (digMap) and select the button

OK (and accept overwriting the setup file). This name is also used as default directory in SCOP projects directory. After creating the image overlay, the corresponding state-switch *digMap...* is placed on the main window below the label *Image Overlays*.

Up to now, only an image overlay was created, but the system does not yet know the image data file itself. By default, the file `digMap.pyr` is searched. This file contains a description of the image pyramid with the names of image data files etc. As this default pyramid description file was delivered when installing SCOP++, the image data file need not be imported in this example.

Now, open the narrow window *digMap* by clicking at the state-switch *digMap...* with the right mouse button. Select the type of the image in the selection *Image type* (*digitized map*). Clicking at the state-switch *digMap...* with the left mouse button and selecting *display* from the pop-up menu will display the map on the main graphics panel. The entire map is too large to be displayed in its original resolution. Instead, a higher level of the image pyramid (with lower resolution) is displayed. That is why some pixel seem to be missing. When zooming into a small part of the map, the original resolution will be displayed.

The map is not shaded, so we will create a shaded view from our DTM and add it to the image. For that purpose, we open the button *TerrainModel...* of our model overlay and switch the state-switch *Shade...* to *display*. First, the model will be interpolated, then, the shading will be derived and finally - as the image overlay is displayed, too - both raster graphics files will be merged. Merging the raster graphic files in the original resolution of the map would need quite some time. The user is asked to accept the mixing result in a lower resolution. Select the button *Yes*. The resulting map gives a much better spatial impression than the unshaded one.

Now, we want to display a mixture of shading, Z-coding, and we want to overlay some of the layers of the digital map. We temporarily switch off the map by changing the state of the state-switch *digMap...* to *off*. Then, we derive a Z-coded view by changing the state of the state-switch *Z-code...* in the window *TerrainModel* to *display*.

In order to set some layers of the digital map to be transparent (i. e. not displayed), we click at the state-switch *Color table...* in the window *digMap* in order to open the window *Palette of image digMap*. In this window , the palette of the image (i. e. its color table) is displayed. Without explaining much, we type in the text field *Transparent color numbers*

```
1, 6
```

(1 is the white background and 6 the light green forest areas) and close the window by selecting the button *OK*. Finally, we change the state of the state-switch *digMap* to *display* again. Only the layers brown (contour lines of the map), dark blue, black, dark green (forest boundaries) and light blue (the lake) are displayed. These layers (e. g. the lake) hide shading and Z-coding, whereas in other areas, only the mixture of shading and Z-coding is displayed. The image you have generated should be the same as in figure 13.2 in section 13.3.

Part III.

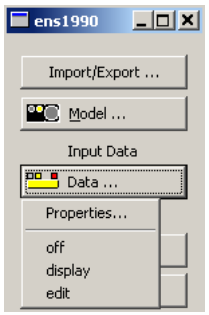
Application Guide

5. General Concepts

5.1. Special Features of the GUI

SCOP++ incorporates some special GUI widgets which might be a short description for the user.

5.1.1. State-Switches



State-Switches are non-standard buttons of the GUI (Graphical User Interface) with a light to indicate the state of the object represented by them. They are in the first line to allow switching this state by the user. But also, state-switches may have a window beneath them; this is indicated by three dots appended to the button's label. Seen from a viewpoint, in which a state-switch denotes an abstraction of an object (like the state-switch *Data...* denotes *the data* in a SCOP++ project), the window beneath it is for interfering with its *properties*.

The figure beside shows a window with various state-switches . In this figure, state-switch *Data...* has the standard form: it carries a light, and, as the dots indicate, it has a window beneath it. The large light shows the selected state while the small lights represent all possible states.

Left-clicking a state-switch will result in a pop-up menu to appear. The possible entries for a standard state-switch are:

- *Properties...*: when chosen, the window beneath the state-switch will be displayed on the screen; it is considered as a means to specify *properties* for the object represented by the state-switch .
- *off*: the first possible state of the switch, usually meaning that the object is passive, shut off, contents not displayed, and the similar.
- *display*: the first active state; when chosen, the object becomes active, and usually its data or product will be displayed (in general at the main graphics panel).
- *edit*: a state allowing to edit the displayed contents belonging to the object.
- *freeze*: (state-switch *Data...* does not have this state) will prevent the contents displayed to be updated on any relevant user actions (e.g., prevent re-computing and re-displaying iso-lines or hill-shading on every single data editing action).

These entries may have other names and other meanings in special cases. In such cases the lights on the state-switch do not have the standard rectangular shape but round lights are used.

Right-clicking a state-switch will directly open the window beneath it – an action identical to the one on choosing *Properties...* in the pop-up menu .

The light on a state-switch may blink (i.e. it may appear as flashing light) to indicate some (lengthy) process.

5.1.2. State-Switches for Views of Secondary Models

This combination of widgets is used to indicate the need to have a prerequisite calculated first (e. g., a secondary model). It is a combination of an ordinary state-switch (cf. section 5.1.1) and an additional round light left of it. The color of the round light corresponds to the calculation-state of the prerequisite: gray means *not calculated*, the blinking light indicates *calculation in progress*, and a yellow light means *available*. The 'ordinary' state-switch is responsible for the visualization and the same remarks as outlined above are applicable here. So, if you set the state-switch to state *display*, first the widget on the left will start to blink, indicating the derivation of the prerequisite, will finally become yellow, and then, the rectangle light will blink, indicating the derivation of the visualization.

The left part of the state-switch does not handle any mouse clicks. All mouse actions have to be done on the right side of the state-switch.

5.1.3. Graphics

Graphics panels in SCOP++ allow displaying both raster graphics (images) and overlaid vector graphics. Even multiple images may be displayed (see section 5.3.3). Details of graphics panels and data editing may be found in section 8.

5.1.4. cmL/cmF

SCOP++ is capable of running in a *batch mode*. The entire GUI of SCOP++ (except the buttons of the graphics panel) may be controlled not only by mouse actions but only by commands.

cmL stands for *command Language*, and *cmF* for *command File*. *cmL* allows for controlling processes from a special window, and *cmF* allows for writing procedures on command files, consisting of *cmL* commands.

First steps on how to control the GUI from the *command window* are already described in section 4.6. Open the command window via the drop-down menu <*cmL*>/*cmL*. All actions the user usually performs by clicking with the mouse or by entering a text via the keyboard can be done also via *cmL* either from the command window or from command files.

A detailed description of the syntax of the command language is given in section 7. For reading how to start SCOP++ in *batch mode*, please refer to section 6.4.

5.2. The Concepts of SCOP Classic

In this section the concepts concerning algorithms and data processing inside SCOP++, which stem from SCOP Classic, are described. For more details see section 17.2. Some parts of this chapter are just a repetition of this section but are repeated here due to their importance.

5.2.1. Objective Target

The objective target is to create the Digital Terrain Model (DTM) using all available input data of the surface (3d point clouds and lines) to suite as well as possible the products to be derived of it. For this purpose filtering of accidental errors in input data is applied and eventually a smooth surface model is generated. The products of this model are derived as requested by efficient and simple methods *without* further amendments.

An example of the above is contour line (isoline) derivation: contour lines as derived of the SCOP DTM, provided that the resolution of it has been chosen properly (seesection 17.8), will not be smoothed, lines will not intersect, and adjacent lines will present a harmony corresponding to the DTM surface.

5.2.2. Algorithmic Concept

SCOP involves a high accuracy DTM interpolation by the method of linear prediction. This is an interpolation technique via summation of surfaces, i.e. the cumulative surface consists of a sum of elementary surfaces around the reference points. The elementary surfaces are rotating bell curves, which allows a statistical analysis. This analysis permits a qualified filtering and smoothing eliminating accidental errors. Essential component of this strategy is the involvement of break- and form lines (see section 5.2.3), which are necessary for a DTM of high quality.

For the description of the terrain surface SCOP uses a subdivision into rectangular interpolation areas of constant size (*Computing Unit: CU*). The surface in each rectangular piece is described by a mathematical function. Discontinuities in the surface are modelled by break lines. In those units separated by one (or more) line(s) the use of more than one function is necessary. The mathematical function or functions are then used to derive a rectangular grid of heights, the DTM grid, providing the discrete description of the surface. The DTM grid replaces the interpolation function, and is to be considered as a discrete form of it. In addition vector-type reference data are integrated with the grid, in order to be able to represent them exactly (rather than just with the accuracy of the DTM grid's resolution); this yields a *hybrid DTM structure* – a specialty of SCOP.

The two methods for the derivation of a DTM – Classic Prediction and Adaptable Prediction – apply the same algorithm. The differences can be found in subtle details. Adaptable Prediction should be applied, if the generation of a simply connected DTM (i.e. a DTM without 'holes') is an important aim. As mentioned above, the Adaptable Prediction is capable of handling very different data densities. In Classic Prediction, on the other hand, a more detailed control of special parameters is possible (e.g. the treatment of

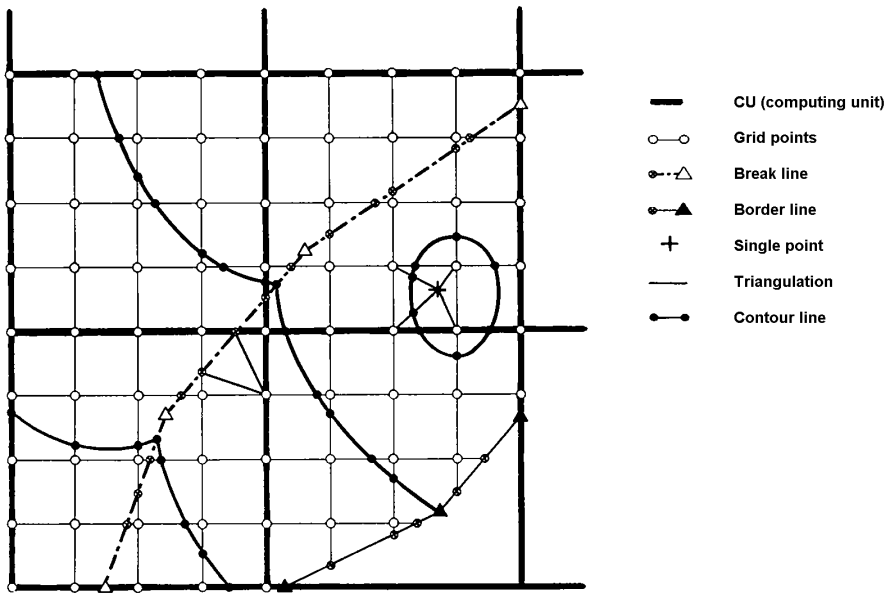


Figure 5.1.: Hybrid DTM Structure of SCOP++

break lines and form lines). With Classic Prediction the generation of a detailed protocol file is possible, too.

In simple cases with a homogenous density distribution in the complete area, and no contradiction in the line data (e.g. crossing break lines without an intersection point) the usage of Classic Prediction and Adaptable Prediction result in similar DTMs.

5.2.3. Primary Data

This section describes the different categories of data serving as source for the interpolation of digital models.

The data used as primary data can be classified

Geometrical		Algorithmic relevance	Further classification
Point		Bulk points	profile points, terrain points, off-terrain points
		Single points	depressions, spot heights
Line	open	Border lines	with/without height
	closed	Break lines	
		Form lines	

These different classes have to be reflected somehow in the data format in order to be passed on to the programm. This can be accomplished by adding an additional information. Two examples for this are:

- Layer information of DXF
- Winput data format (see appendix B)

Bulk Points

Bulk points are the terrain points without specific relevance to the algorithm. They can stem from profile measurements, raster measurements and the similar. Filtering of accidental errors in this primary data type is one of SCOP's benefits.

Single Points

Unlike bulk points, single points - also denoted as spot heights - are data carrying "vector type" (individual, specific) information. Usually, these are points to be labelled with their elevation on the map. Most often, they are located on hill tops, or at the bottom of a depression. Spot Heights can be given a special filtering value for interpolating the DTM. By defining small filtering values, the DTM surface can be forced to approach these points very closely.

Border Lines

Border lines are to mask off areas of interest. For instance, to avoid plotting of contour lines on streets and buildings, these should be delimited by corresponding border lines.

Border lines should:

- either surround the entire area of interest (for a good quality of the DTM at its edges, it is necessary to have data outside the surrounding border line),
- or end in both directions outside the DTM area,
- or be represented by a closed line to indicate holes in the area.

Different areas masked off by border lines must not overlap.

Break Lines

It is helpful to bear in mind that the DTM grid is representing the interpolating surface; and that the last is a generalization of the terrain surface. The interpolating surface is subdivided into continuous areas by break lines.

The algorithm realizing this subdivision is complicated by two circumstances:

- that, in order to enable the above subdivision, break lines have to be linked by the program so to form a network;
- that interpolation is performed sequentially for (overlapping) computing units ('CU').

Break line networking, and surface subdivision (into interpolating areas with a surface assumed to be continuous) are done for every CU separately.

Of the above, important rules follow for recording break lines:

- a break line is subdividing a CU into separate interpolating areas only if
  - the break line is closed,
  - both ends of the break line are beyond the limits of the CU,

  - or, as a result of networking, the break line has no free end within the CU.

  Therefore,

  - interpolation is not influenced by very short break lines standing isolated.
  - open ends of break lines should be extended somewhat into the area with continuous surface. Otherwise artifacts may occur in the products derived of DTM.

- Networking a complicated system of break line endings with more than one node within a single rectangle of the DTM grid may cause structural problems.

An automatic densification of break line points may be requested by the user. Break line points, and intersections of these lines with the DTM grid are computed (e.g. predicted, or, if so requested, interpolated linearly along the break line), and intermeshed with the DTM structure. For break lines, a separate filtering value can be defined (effective in linear prediction with the Bell Curve as covariance function).

Form Lines

Form lines are sometimes explained as "round break lines". Intersections of form lines with the DTM grid are computed in ways similar to those for break lines, and intermeshed with the DTM grid. An automatic densification of points along form lines can be requested by the user. For form lines, separate filtering amounts can be defined.

Form lines do not subdivide computing units to separate interpolating areas.

Grid Data

Data acquired (or computed by some other program) in form of a regular (rectangular) grid can be accepted as such. Grid data may cover just patches of the area, and files with such data may contain other data of any type (e.g. break lines). When taking over grid data as the DTM raster, only those points missing on the data set will be interpolated.

Grid data can be input in various formats (e.g WINPUT - see appendix B).

5.2.4. Density of Primary Data

Data density should be, at least as an ideal, such as to describe the terrain surface with a predefined accuracy. This criterion yields densities varying locally with the constitution of the surface. Spectral analysis is a method to define data density and distribution using the above criterion; progressive sampling aims at data acquisition corresponding to it.

In practice, however, the grid step of the DTM is usually the starting point of considerations on data density. Grid step may be predefined in customer contracts, or defined by previous experience with similar cases. If a map with contour lines (isolines) is requested, a further simple consideration can be taken into account: a grid step of some 2 to 4 mm on the map is necessary to get contour lines acceptable in a cartographic sense.

The average distance between terrain points should not exceed 2 to 3 times the grid step; if this average distance exceeds 10 to 15 times the grid step, unacceptable DTM will result.

Circumstances limiting the maximal distance between terrain points are:

- in areas without terrain points larger than some 5x5 computing units the DTM will not be computed (a CU, with a size defined automatically, is often just 2 grid step wide);
- in areas without terrain points, the interpolation surface will closely approach a tilted (or, in special cases, horizontal) plane. It is, however, no plane in a strict sense. Therefore, for the sake of having predictable results, one should record at least some terrain points in areas without changes in the terrain surface, as well.

Data should express every terrain form with some redundancy; this allows for filtering in interpolating the DTM.

Along break-, form-, and border lines, it is sufficient to record as many points as necessary to replace curves of these lines by polygons.

5.3. The Concepts of SCOP++

SCOP++ is based on algorithms of SCOP Classic. Many background computations are performed by modules of SCOP Classic. Additional concepts are described in the following sections.

5.3.1. Lazy Processing

In SCOP++, no computations will be performed unless

- the results are needed for any active product (e.g. a displayed hill-shading),
- the results are prerequisite for some other processes switched active (e.g. the DTM is prerequisite for a hill-shading),
- or, in special cases, unless there is an explicit request on behalf of the user (an example of this is provided via the button *Interpolate* in the properties' window to state-switch *Model...*).

This principle carries the name *Lazy Processing*.

As a consequence of the Lazy Processing paradigm some (lengthy) processes are delayed and may become necessary when the user does not expect it. For instance, if some state-switch (e.g. the one for hill-shading) is set to an active state (*display*), and the DTM has not yet been interpolated (or the DTM was interpolated with other than the current parameters), DTM interpolation will be performed automatically before deriving the hill-shaded view.

5.3.2. GUI Design

The overall design of the graphical user interface (GUI) is influenced by the following concepts:

- SCOP++ starts with one main window which is dominated by the main graphics panel serving as display area of all planimetric data and derived products. The co-ordinate range accessible by scroll-bars is limited to the area for which data is available (except if "zoom+" mode is selected, see section 10). The main window of SCOP++ can be stretched to fit best to the desktop of a user.

- The main window has some drop-down menus for general actions like opening, saving, closing a project, adding and dropping overlays to/from a project, etc. The drop-down menu *Tools...* offers some general tools for one of the overlays.
- Left of the main graphics panel, the button for the general mode (see section 5.3.4) and the buttons of overlays (see section 5.3.6) are placed.
- On the right side, buttons which are independent of one specific overlay are placed (e.g. the buttons *Frame...*, *Limits...*).
- On the bottom of each window, a status bar with a context specific help text depending on the current position of the mouse is situated.

5.3.3. Raster Graphics Mixture

One of the key features of SCOP++ is the central graphics area in which all products are displayed. Many different vector graphics datasets can easily be displayed simultaneously. Multiple raster graphics (images) would hide each other.

For this purpose SCOP++ has a sophisticated management of displayed raster graphics files. If mores image overlap, SCOP++ knows two strategies of displaying all of them:

1. Creating a mixture of these images for the overlap area: Different mixing rules are defined for various combinations of image types (e. g. adding the shading gray value minus 128 to the intensity value of a Z-coded image or calculate the average value of two grayscale orthophotos etc.). As the mixing rule depends on the semantic type of an image, it is important to specify this type for image overlays (cf. section 13.3 for details about mixing rules).
2. Using transparency channels: for images which are represented as ”palette images”, some channels of the image may be set transparent(cf. 13.9). In that case, only those pixels which are not set transparent are displayed and hide the other image(s) whereas ”behind” transparent pixels the other image(s) can be seen.

If the operator switches on an image (either an image overlay or a raster graphics view of a model overlay) and it has an overlapping area with (an) already displayed image(s), the system chooses automatically the proper strategy of displaying both. If it is not possible to create a mix (e. g. if two true color images are to be displayed), the previously displayed image(s) will be switched off automatically.

Mixing two images usually takes some time especially if the two images do not have congruent pixels. In that case one of the images must be resampled to the grid of the other one. Thus, we recommend to select identical pixel size for these images.

The raster graphics management in SCOP++ has some useful applications. For instance, it allows to

- overlay a hill-shaded view of the terrain over a digital map. In this way,maps without shading get a more realistic impression.
- visualize discrepancies between overlapping orthophotos created from neighboring photographs.
- overlay only rivers and lakes of a digital map over a mixture of Z-coded view and shading (cf. figure 13.2).
- etc.

5.3.4. General MODE

Applying General MODE properly will, on the one hand, allow for more effective work with the program, to avoid annoyance with superfluous automatism, and, on the other hand, to specify the resolution of SCOP++ products as needed.

In the upper left part of the main window there is a state-switch *MODE*. Left-clicking it brings up a pop-up menu of three modes: *Final, Screen, and Freeze processing*. The color of the light on state-switch *MODE* reflects the selected mode.

Final (gray light) is for creating final versions of the different products of SCOP++. In this state the pixel size of different products can be specified by the user. This might be of crucial importance in certain cases, such as deriving high resolution raster graphics for external use.

Screen (yellow light, the default) causes graphics files to be derived so to correspond to the resolution of the screen. As long as this MODE remains active, specifications such as of pixel size will not be available.

Freeze processing (blue light) will prevent automatic re-computation of graphics displays on individual changes of data or of parameter values all over the SCOP++ system. This mode is very useful, e.g., when editing data, where re-interpolating the model and re-computing all active displays on every single change could become cumbersome. This might in some cases also be true for working with limits (see section 5.3.13).

5.3.5. Projects

SCOP++ is designed to work with exactly one project. The reader of this chapter should already have read the section 3.1 in part II.

A project is the superior entity in SCOP++. It comprises zero to about thirty overlays (this number is restricted due to performance and resource limitations).

Settings which have project scope are:

- Limits: Areas of interest defined in the window *Limits* (see section 5.3.13) have project wide validity. They refer to all overlays within a project.
- Scale: The scale intended for the derived products also has project wide validity (see section 9.2.1 and section 9.2.8).
- Protocol options: The protocol options (see section 9.4.4) are valid for the whole project.
- Project settings: The protocol settings (see section 9.2.8) are valid for the whole project.

5.3.6. Overlays

SCOP++ is designed to work with several different overlays simultaneously. The reader of this chapter should already have read the section 3.2 in part II.

5.3.7. Model Overlays

Model Overlays are carrying different values along the elevation (Z) axis, and in special cases just clouds of points; some examples for model overlays are

- elevation models,
- accuracy models,
- check points,
- off-terrain points,
- slope models,
- exposition models,
- soil quality models,
- models representing local susceptibility to soil erosion,
- models belonging to different epochs,
- etc.

A SCOP++ project can contain more than one model overlay. Basically model overlays are thought to be layers of different types of models (DTM, digital slope model, ...) covering the same area of interest. Apart from that model overlays of the same type can also cover different areas. Subdivision of larger areas of interest (project) into smaller parts (model overlays) can be done in that way. Furthermore it is possible to have model overlays of the same type covering the same area.

An example for the latter case can be a flood simulation. For a flood simulation a terrain model and a model of the water level (both being digital models in terms of SCOP++) are required. From these models a difference model can be calculated (terrain model minus model of water level). All these three models are separate model overlays carrying the same type of information (elevations) and covering the same area. On the basis of the central graphics panel it is possible to overlay different products derived from these models. For example it is possible to visualize a Z-Coding (see 12.8) of the difference model, showing the flooded area in different colors of blue, together with the contour lines (see 12.6) of the original terrain. As you can see the concept of model overlays allow different applications.

There are three types of model overlays:

Data only: Such overlays consist only of primary vector-type data. No models can be derived from these data, but they can be displayed in conjunction with other model and image overlays. Typical examples are 2D data like roads, rivers, borders of cities or countries and the similar.

Model only: Overlays of this type contain already pre-calculated digital models. These models are separated from the original data and therefore cannot be changed. Nevertheless, views (see 12.6, 12.7, 12.8) can be derived from them. Typical examples for model-only overlays are slope models or difference models of two DTMs.

Both: Overlays of type both contain digital (elevation) data from which models are to be derived by SCOP++. The data can be edited (see 12.5) and the parameters for the interpolation of the model (see 12.3) can be changed to achieve best results concerning the derived model (DTM).

File based processing: This option is only available with model overlays of type *Both*. Activating it circumvents the usage of the internal database by using the input files straight away. For details see section 5.3.11.

Adding and dropping overlays is managed by the drop-down menu *Overlays*. Please refer to the part IV, section 9.3 for more details.

5.3.8. Image Overlays

Image overlays can be added to a project in SCOP++ in order to display digital imagery. Typical images which may be of interest to display in a SCOP project include:

- digital orthophotos,
- rectified satellite imagery,
- thematic maps,
- vector maps converted to raster graphics,
- views of model overlays which have been exported to image overlays (e. g. a hill shaded or elevation coded image).

Adding images to a project can have several benefits. Digital images (i. e. raster graphics files) are displayed together with (behind) vector graphics. The image (e. g. an orthophoto) can be viewed simultaneously with input data such as break lines or with derived data such as isolines. In this way, the data can be checked and errors in the data may be detected.

If more than one raster graphics files are to be displayed simultaneously, the one added at last will usually hide the others. Nevertheless, the system SCOP++ tries to create a mixture of these raster graphics files. If successful, this mixture is displayed and the user has the impression of displaying multiple images simultaneously(cf. 5.3.3.

Image overlays are described in detail in the reference manual in section 13.

5.3.9. Database

For the following tasks SCOP++ uses an internal database (see also section 3.3):

- Storage of input data,
- Translation of coding information
- Data access.

Storage of input data

Data read from one or more input data files are stored in a database table. For each project one database table is created. This table is physically stored as file with the extension "`.TOP`" in the projects root directory. This file should only be accessed by means of SCOP++ and must not be edited or deleted manually!

The use of a database has one important consequence: Once a project has been created, it cannot be renamed any more. Although it is allowed to move it to another directory (i.e. to define another path), the name of the project itself (e.g. the name of the project file `*.spr`) must not be changed.

Table 5.1.: Classes of input data and their Winput Codes

Data class	Description	Winput Code
Bulk data	mass points, profiles, contour lines, ...	10 .. 30
Single points	depressions, peaks, highs, lows, ...	31
Form lines	"soft" ridges, ...	40, 41
Break lines	ridges, shorelines, dams, ...	50 .. 56
Border lines	lines to mark off areas of interest	60 .. 67

Translation of data coding

Not only the geometry of the input data but also the coding information of each data object is stored in the database. The coding information describes the meaning and the geometric type of a data object. Since it is stored in a different way in each data format (eg. DXF: Layer, Entity), coding information is translated during data import into an abstract coding scheme of the database using so called *Code Conversion Tables*. The concept of storing and translating coding information in SCOP++ is explained in detail in the next section.

Access

All data displayed in the main graphics panel, or used for interpolation is accessed via the database. Efficient communication policies make this approach fast enough. Among the great benefits is the unique response also on specific spatial requests and the inevitable consistency of data.

5.3.10. Data Coding

The following section contains information how coding information is managed in SCOP++. To take full benefit from SCOP++ it is important to understand the underlaying concept of data coding. So please read this section carefully!

As described in section 5.2.3 SCOP accepts different classes of input data. In classical SCOP data are passed to the various calculation processes in SCOP Winput data format (see also appendix B), where the coding information is stored in a two-digit numerical code.

Table 5.1 gives an overview about the different classes of input data and their appropriate Winput Codes.

This is a very simple coding scheme, appropriate if data is collected for DTM production only. But in many cases, 3D topographic data are coming from other systems (CAD, GIS, map production, ...) and are carrying a more detailed coding structure.

As an example a CAD drawing may contain shorelines, dams, rivers, ridges, etc., all in different layers. To compute a DTM, these data should be passed to SCOP++. Regarding

Table 5.2.: Object types and their meaning

Objecttype	Meaning
CLUSTER	scatter-plot, regular or irregular distributed points
SYMBOL	single points like depressions, peaks, ...
POINT	single points (no special meaning for the surface)
LINE	open line string (open polyline)
AREA	closed line string (closed polyline)

the simple coding structure from above, all the mentioned features have to be coded as break lines (WINPUT Code 50) and the detailed coding structure of the CAD drawing gets lost.

To avoid this loss of coding information SCOP++ uses an abstract coding scheme for storing the data's coding information in the database. This coding scheme is realized by the attributes *Featurecode* and *Objecttype*, which are stored together with the geometry for each data object.

- *Featurecode* describes the meaning of the object (eg. river, dam, coastline, etc.). Although there is a set of predefined feature codes, the user may define his own ones.
- *Objecttype* describes the geometric type of the object. The list of valid object types is fixed and contains the items CLUSTER, SYMBOL, POINT, LINE and AREA.

This organization is independent from the import data format. For this reason it possible to manage input data coming from different file formats within a single overlay without loosing coding information.

Tables 5.2 and 5.3 contain the complete list of predefined *Objecttypes* and *Featurecodes* as well as their meanings, being part of any SCOP++ installation.

The predefined *Featurecodes* are kept close to the Winput codes (see also B). To obtain a more detailed coding structure own feature codes may be added to the list of predefined codes (see section 9.4.2 for details).

The conversion of the coding information is controlled by the so called *Code Conversion Tables* (in the following the term *Conversion tables* will be used). Conversion tables are database tables which are accessible from any SCOP++ project for importing (and exporting) primary input data.

The definition of the structure of a code conversion table and its contents depend on the foreign data format. For this reason there is an own *type* of code conversion table for each supported data format. The following columns definitions are used.

As shown above all conversion tables share the columns *Featurecode* and *Objecttype*, as they represent the coding information as stored in the database. Furthermore each conversion table contains one ore more columns representing the coding elements of the specific data format.

Concerning coding of data it is important for experienced users of SCOP classic as well as for new SCOP++ users to think in terms of feature codes and object types rather than

Table 5.3.: Feature types and their meaning

Featurecode	Meaning
PROFILE	profiles (x constant, y constant, ...)
ALIGNMENT	(measured) alignments
CROSS_SECTION	(measured) cross sections
CONTOURLINE	(measured) contour lines
RANDOM	randomly distributed (mass) points
SPOTHEIGHT	single points like peaks, highs, ...
FORMLINE	structure lines (smooth break lines)
BREAKLINE	break lines like ridges, dams, ...
BREAK_BORDER_OMIT_RIGHT	break lines also serving as border lines (omit area to the right)
BREAK_BORDER_OMIT_LEFT	break lines also serving as border lines (omit area to the left)
BORDERLINE_OMIT_RIGHT	3D border lines (omit area to the right)
BORDERLINE_OMIT_RIGHT_NOZ	2D border lines (omit area to the right)
BORDERLINE_OMIT_LEFT	3D border lines (omit area to the left)
BORDERLINE_OMIT_LEFT_NOZ	2D border lines (omit area to the left)
OUTER_BORDERLINE	outer borderlines enclosing the area of interest
INNER_BORDERLINE	inner borderlines defining exclusion areas
SINGULAR	off terrain points
SITUATION	elements of the situation
SITUATION1	— ″ —
SITUATION2	— ″ —
SITUATION3	— ″ —
SITUATION4	— ″ —
SITUATION5	— ″ —
SITUATION6	— ″ —
SITUATION7	— ″ —
SITUATION8	— ″ —
SITUATION9	— ″ —
PLANIMETRY	— ″ —

Table 5.4.: Column definitions of Conversion tables

Winput	DXF	ArcInfo Generate	XYZ	Kotenband
Featurecode	Featurecode	Featurecode	Featurecode	Featurecode
Objecttype	Objecttype	Objecttype	Objecttype	Objecttype
Remark	Remark	Remark	Remark	Remark
Winputcode	Layer	SFT		Keycode
	Entity			

in terms of Winput codes. Nevertheless, since the model interpolation and other server processes in SCOP++ are still based upon data coded in Winput, it is necessary to be familiar with this data format.

Figure 5.2 shows the flow of input data within a SCOP++ project, with focus on the coding information:

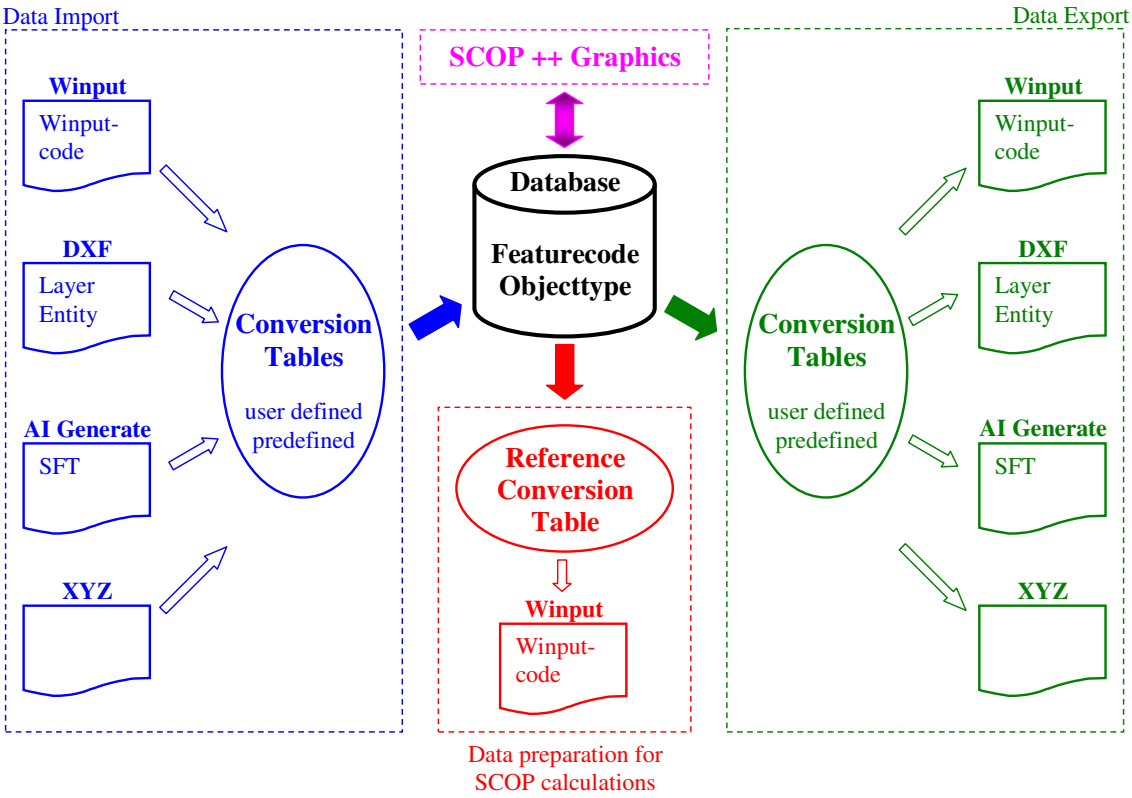


Figure 5.2.: Data Flow within SCOP++

Consequences:

- During data import the conversion tables are used to control the translation of the coding information from the data format specific form (eg. DXF: Layer, Entity) to the common representation (Featurecode, Objecttype) in the database.
- During data export the translation is done in the opposite direction, again using the conversion tables.
- The preparation of input data in Winput data format, to be used in the various server processes in SCOP, is done using a special conversion table (Reference table). See below for details about the meaning and use of the Reference (conversion) table.
- The SCOP++ graphic directly interacts with the database. There is no need to translate the coding information.

Figure 5.2 indicates, that there are three different categories of conversion tables. The differences only concern the field of application within SCOP++ but not their structure.

- *Predefined Conversion Tables:* For each supported data format there is one predefined Standard Conversion table. These tables tables cannot be changed by the user. Stan-

dard Conversion tables can be used for data import and export and they are set up in a way, that files produced by the various SCOP application modules (Isolines, etc.) can be read in directly in SCOP++. Section 12.2 contains a complete reference of the predefined Standard Conversion tables.

- *User defined Conversion Tables:* If the standard tables do not fit the requirements, the user may create and edit his own conversion tables. This is first of all necessary for DXF format, where different files my contain different layer structures. Another valuable application of user defined conversion tables is, if the predefined feature codes are not detailed enough. In this case, the user can add his own feature codes and can use them in the user defined conversion tables. Details about adding or changing feature codes and about the creation and manipulation of conversion tables can be found in section 9.4.3.
- *Reference (Conversion) Table:* For the preparation of data in Winput data format to be used in various SCOP server processes, a specific conversion table - the so called *Reference Table* (CVWNPREF) - is used. This table may be edited by the user but cannot be deleted. For every combination of the coding pair Featurecode/Objecttype the reference table must contain the appropriate Winput code, so that the SCOP server processes can handle these data. Due to that fact, every time a new Featurecode/Objecttype combination is introduced in one of the user defined conversion tables, an appropriate Winput code must be entered in the reference table, otherwise those data would be ignored by SCOP. At least for this reason, the user should become familiar with the Winput data format.

5.3.11. File based processing

The basic principles of data administration in the internal database and handling of coding information are described in the previous sections. The use of a geo-database allows handling of data with complex coding structure within a single SCOP++ project.

Anyway, in some cases this advantage may not be utilized. This becomes especially true for point cloud data (e.g. from LIDAR of automatic image correlation). In this case it is more convenient to circumvent the internal database and to use the input files straight away. For theses purposes SCOP++ provides a *File bases processing mode* in addition to the standard database mode.

In *File based processing mode*:

- input files are not imported into the internal database.
- (multiple) input files are directly converted into a single Winput file (`overlay_.all`) containing all overlay data. This file serves as data source for all subsequent operations like model interpolation, data display, etc.
- The list of supported data formats is restricted to XYZ (ASCII and binary), Winput (ASCII and binary) and LAS.
- several other overlay related files like the `_L_`-File (to be used in the Limits window) are derived during data import.

Please note, that there are no restrictions concerning program functionality when using the *file based processing mode*. Rule of thumb: The database mode is better suited if the

data are complexly structured (eg. different types of break lines, structure lines, ...) whereas the file based mode is more convenient for data with simple coding structures (eg. point cloud). Furthermore the file based mode simplifies batch-based work flows. Thus, A standard scenario for using the file based mode is importing one or more data files, interpolating the DTM, deriving products like isolines, hill shadings or profiles, exporting the DTM and subsequently perform the same tasks with different input files.

5.3.12. Concurrency

SCOP++ is designed as a concurrent system. This means that the system allows apparently simultaneous handling of different tasks. This is also the cause why the user interface isn't blocked during time consuming server processes and even further processes may be started.

SCOP++ manages the synchronization of the resources (e.g the DTM file) and the observation of sequence. So, if the user requests new contour lines to be derived, it is first checked, if the DTM has to be updated, potentially awaiting the completion of this task and only if this hasn't produced any errors, the contour lines will be derived. SCOP++ even checks if during computation time the parameters (of the contour lines or the DTM) have been changed and will automatically trigger a re-computation.

Processes are started either directly by the user or indirectly (as a consequence of the lazy processing principle, see section 5.3.1). In the second case, the user has often not expected to start a progress. For most of the processes he has the possibility to interrupt them.

Lengthy processes are registered at SCOP++ in the window *Background computation* where they can be *interrupted*. A process which is capable of estimating its progress, will additionally display a progress bar in the bottom of this window . It is opened automatically if a registered process starts running. It will be closed automatically when all processes are terminated. However, the user can close the window and open it by selecting the drop down dialog *Server processes...* from drop-down menu *Project*. Details about this window can be found in section 9.2.9.

The fact that a process is running in background can be seen either in the window *Background computation* (described above) or by a blinking state-switch (see section 5.1.1).

5.3.13. Areas of Interest

At the lower right corner of the main SCOP++ window there is a button *Limits...* Clicking at it will open a window *Limits* with a graphics panel and different controls in it (Fig. 5.3). If there are any model overlays carrying data included in the current project, the graphics panel will display an overview of those data, to contain all lines and a little percentage of bulk data.
The window *Limits* is to control areas of interest for different purposes. These areas are identical for all overlays of the project.
Areas of Interest are rectangular areas digitized in the graphics panel, or entered and/or edited in the window *Edit...*. They serve four purposes (group *Limits for*):

- screen – to control the main graphics panel
- model – to delimit the DTM to be interpolated
- views – to delimit different views to be derived of the DTM
- data – to delimit the area for displaying and exporting data.

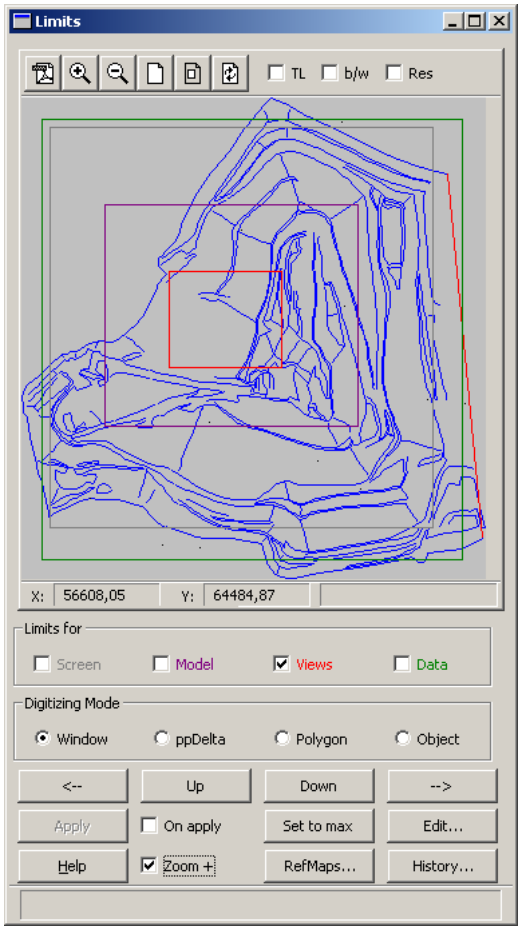


Figure 5.3.: Window LIMITS

Screen and Data limits are independent of Model and Views limits. Views limits depend on model limits, because views can only be derived for areas where a model exists. When however limits defined for model and views are not overlapping, no valid view can be derived.

Please note that the color of the radio buttons in group *Limits for* is identical with the color of the corresponding rectangles in the graphics panel.

Editing actions (such as digitizing a new window) will influence all categories (purposes) currently selected in the selection *Limits for*. For more details please refer to section 10.

5.3.14. Help System

In most of the application's windows you will find a button *Help*. Hitting this, the relevant section of the manual will be displayed. If there cannot be found a specific section, solely the first page of the manual will be displayed.

6. Working with Projects

6.1. Workflow of Projects

In this section, typical workflows of projects are described. The operator is already expected to have got first experience with SCOP++. Some examples are described step by step. The examples become more and more specific. Suggestions are made on how to handle large data sets with minimum computation time in the phase of searching proper parameters.

The steps of the workflow are described by only a few words. Details on how to perform the steps can be read in the part IV (for each step, the corresponding section in the reference manual is given).

The examples are explained for a new project, but using the data of the sample project *albis* which is included on the installation disk. You should find the data in the directory `albis` in your SCOP-projects' directory. If not, you will also find it in the programs-directory.

Example 1: First, we imagine a typical project. Given a set of *3d points* the aim is to derive a plot of *contour lines* (as file of format DXF).

1. Create a new project (via drop-down menu *Project/New...*, cf. 9.2.1). A directory on your file system is created and the window *Add model overlay* is opened automatically.
2. Add a model overlay of type "Both"(via drop-down menu *Overlays/Add model overlay...*, cf. 9.3.1). A directory for the overlay will be created.
3. The data (the file of the 3d points) must be imported. We recommend to copy the file containing the data into the overlay directory (copy the file `terrainModel.wnp` from the overlay-directory `terrainModel` of project *albis*). Opening the button of your model overlay will automatically open the window *Import data* (cf. 12.2.1). Select the file for importing. The data will be stored in the database and can be displayed (via setting state-switch *Data...* to *display*, cf. 12.5).
4. In order to display the contour lines (i. e. the iso-lines of elevation), the state of the state-switch *Isolines...* is set to *display*. Before the contour lines can be calculated, the DTM has to be derived. Thus, the state-switch *Model...* starts blinking (and the window *Background computation* appears). When DTM interpolation has finished, the state-switch *Iso...* starts blinking indicating that deriving contour lines of the DTM has started. Finally, the derived contour lines will be displayed at the main graphics panel.

5. The displayed contour lines are stored somewhere in a temporary file (in some system specific data format). In order to save the result in a user defined file (and in a user defined data format), the contour lines have to exported. This is done via button *Import/Export...*, selecting *Views* and button *Export...*, cf. 12.2.7).
6. Exit the program (via drop-down menu *Project/Exit*). Note, that it might happen that SCOP++ does not shut down immediately. This may be the case when exporting the contour lines still is in progress. In that case, the system will wait for the termination of running processes and shut down correctly afterwards.

Example 2: Instead of contour lines, a *color coded map* should be created where different elevation levels are assigned different colors. The map should be produced in *final mode* with a pixel size of 1 m.

1. Open the project of example 1 (e. g. via the list of recent projects, cf. 9.2.3).
2. Switch the Z-coding to display. SCOP++ will recognize, that the DTM is still up to date and need not be interpolated again. Thus, the state-switch *Z-code...* will blink for only a short time and a first result of a z-coded map is quickly displayed. This is because of mode *Screen* (cf. 5.3.4), in which the pixel size is chosen in order to avoid unnecessary calculation time.
3. Now, you can adapt the color palette or other parameters (cf. 12.8). Accepting the new parameters by selecting button *OK* or *Apply* will create new Z-coded maps, again in short time.
4. Now we want to create a map in higher resolution. As preparation, we set the global mode to *Final* ((cf. 5.3.4) and set the state-switch *Z-code...* to *freeze*.
5. In final mode, the pixel size can be adapted. We set pixel size 1 m yielding an image size of 4000x4000 pixels (cf. 12.8.7). Now, by selecting button *OK*, the map won't be re-calculated because of state *freeze*.
6. Finally, the image (with a user-defined file name and file format) is to be calculated in the requested resolution. This is done again via button *Import/Export...*, selecting *Views* and button *Export...*, cf. 12.2.7).
7. Shut down SCOP++ by selecting drop-down menu *Project/Exit*.

Example 3: In this example, an *image overlay* shall be used for checking the result. A digitized map can be added to the project to compare the derived contour lines with the ones of the map. If there are discrepancies, the error has to be searched in the input data. We will learn how to *edit the data*.

1. Open the project of example 1 (e. g. via the list of recent projects, cf. 9.2.3) (or that of example 2 and switch back to *Screen* mode).
2. Display the contour lines (by switching the state-switch *Isolines...* to *display*). As no parameters have been changed, the previously computed contour lines can be used and are displayed immediately.
3. Add an image overlay via drop-down menu *Overlays/Add image overlay...*. Select a proper name for the overlay (independent of the name of the image data file, cf. 9.3.2). A directory with that name will be created.

4. Import the image data file. We recommend to copy the file into the overlay directory (copy the file `albis.tif` from the overlay-directory `digMap` of project *albis*). Opening the main state-switch of your image overlay will automatically open the window *Import image* (cf. 13.4). Select the file for importing.
5. Set the type of the image to *digitized map* (cf. 13.3).
6. Display the image by setting the main state-switch of the image overlay to *display*. You will be asked whether you want the system to calculate an image pyramid. Images for which image pyramids are available need far less time to be displayed. Thus, answer the question with *Yes*. The system will choose appropriate parameters depending on the specified image type. When the calculation of the image pyramid is finished, the map will be displayed.
7. Now, that both the map and our contour lines are displayed, we want to compare them. Unfortunately, the color of the contour lines is similar. So we change the color of our contour lines. This can be done by opening the properties' window of state-switch *Isolines...* of our model overlay (cf.12.6).
8. Suppose you detect discrepancies between the contour lines of your model overlay and the ones of the image overlay. Suppose that these discrepancies stem from erroneous points in your primary data set. We want to eliminate these points. So we switch on the data by changing the state of the state-switch *Data...* to *edit*, cf. section 12.5 (assure that the checkbox *Hide* is not selected for bulk points in the properties' window of the data). Additionally we switch on the display of the Z-coordinate of the points by selecting the checkbox *ShowZ*.
9. If an erroneous point has been detected, it can be deleted from the dataset by the button *Delete Point from Element* (cf. section 8). The DTM will be re-interpolated immediately and new contour lines will be derived.
10. In order to avoid superfluous re-calculations, we make use of the mode *freeze*. This could be done by setting the general mode to *Freeze processing* (cf. 5.3.4). In our example, it is sufficient to set the state of the state-switch *Isolines...* to *freeze*, because the contour lines are the only displayed (active) product. Due to the *lazy processing* principle the DTM won't be re-interpolated if there is no active product requesting it.
11. After several further editing actions, the state can be reset to *display*, which will result in a new DTM interpolation and new contour line derivation.
12. Shut down SCOP++.

Example 4: This example will show some suggestions how to use *limits* and explain an important advantage of *model-only* overlays.

1. We continue the project of example 3 and reopen it.
2. We want to continue editing the data. We want to see the effects of editing the data immediately. Therefore we do not want to freeze processing. In order to guarantee fast recalculation we will restrict the area of interest to the displayed area. We select all four checkboxes *Limits for* (cf.5.3.13), because we want to zoom directly to our area of interest and we want to restrict

- the area for which the DTM is re-interpolated,
- the area for which contour lines are re-calculated,
- and the displayed data (i. e. display only points inside the area of interest).

3. Now, we can define a new area of interest (e. g. by digitizing a rectangle over the displayed map in the graphics window of the window *Limits*, cf. section 10). If the checkbox *On Apply* is not checked, the new area of interest will immediately be applied (otherwise the button *Apply* must be pressed). In our example, the system will ask you whether you really want to set new model limits. The system obviously has some doubt, because there is a large DTM available, and setting new model limits would throw away this DTM and replace it by another which would be covered by the old one as well. Despite, we answer with *Yes*, because we want to continue working with a smaller DTM.
4. Set several times a new area of interest and check the derived contour lines. In case of disrepancies, edit the data. Check the whole area of data step by step. Try also the buttons <–, *Up*, *Down*, –> for moving the area of interest.
5. Finally, when the whole area was checked, switch off the data and reset the area of interest to the maximum area for which data are available by selecting the button *Set to max*.
6. Now, all experiments for deriving a good DTM have been finished. We want to "freeze" the derived DTM and use it for deriving further products (e. g. a hill-shaded map). For that purpose, we recommend to export the internally used DTM-file to a user-specified file name and add a model overlay of type *model-only*. Open the button *Import/Export...*; see section 12.2.6 for details. The new model-overlay will be added to the project.
7. We can drop the model overlay of type *both* as we won't need it any more. This can be done via the drop-down menu *Overlays/Drop model overlay...*. The contour lines of that overlay will disappear from the main graphics panel, because the whole overlay was dropped from the project. However, it can simply be readded again, if necessary.
8. If we want to display the contour lines again, we can derive them from the new model overlay. We can restrict the area of interest again, but the DTM won't be changed.

6.2. Application Strategies

Evidently, a complex system like SCOP++ can be applied in many different ways. It is the aim of this section to help you finding the ways best suited to your projects – but also to your personality and experience.

Two different approaches could be:

- "All-over User-Specifications – Consecutive Refinement" This is probably the better choice for large projects because it may need less computations (provided that user specifications are better than defaults, and that *LIMITS* are applied reasonably).

  This strategy involves:

  - setting General MODE to *Freeze processing*
  - setting parameters on different windows *Properties...* as seen proper, and setting the state of some state-switches to *display* (due to General MODE *freeze*, these local settings will be immediately converted to *freeze* also)
  - choosing *LIMITS* for different purposes to correspond to some characteristic area of interest
  - setting General MODE to *Screen* This will start computation (DTM interpolation if needed, deriving *views* as requested – and displaying them)
  - repeating the above actions so to find better settings for parameters;
  - data editing as needed, with setting and re-setting general MODE at convenience, and also with choosing *LIMITS* correspondingly;
  - and finally, setting General MODE to *Final*, setting pixel size and similar parameters to reflect the resolution of the final product to be derived, clicking at the corresponding OK or Apply buttons , and *Exporting* the products thus derived.

  The advantage of this strategy is that lengthy calculations are postponed and all calculations are done at once (without the necessity of user presence).

- "Starting with Defaults – Trial-and-Error" This is a more easy-going approach, well suited for small experimental projects or for a part of a larger project (defined by using areas of interest, cf. section 5.3.13. For such projects this strategy might prove to be not just simpler but also faster.
  - set the state of those state-switches of interest to *display*. This will immediately start computation applying the defaults as derived by SCOP++; these defaults are based on data distribution and *meta data* available.
  - inspect results, refine parameters where needed;
  - proceed further with data editing as in the first strategy as described above, and end with the final step in the same way.

6.3. Arranging the Environment

How you arrange the working environment, is dependent on the number of projects you are working with, and, to a large extent, on personal preferences.

- Placing a symbol for SCOP++ onto the screen without any command line parameter to it. This is done by the installation program. There will be a dialog to open the project (e. g. Ens) by selecting a project file (e. g. `Ens.spr`) via a file browser; or, if it is one of the recent projects one worked with, then just choosing it in the corresponding list of the most recent projects.
- If some project has been active in SCOP++ earlier, the file `project.spr` as created for it by the most recent run of SCOP++ can be dragged and dropped over the symbol for SCOP++ so to re-open it.

- Similarly, separate SCOP++-icons can be placed on the desktop for all projects. In this case, the program can be started with the project name (including its path) as command line argument.
- system means allow for specifying files with the extension *.spr* to be opened by `scop++.exe`. With this specification, double-clicking any file `project.spr` will start SCOP++ for the corresponding project.

6.4. Run SCOP++ in BATCH mode

As described in section 5.1.4, SCOP++ can be controlled via commands read from a command file. Furthermore, it is possible to start SCOP++ in a special BATCH processing mode. The intention of the batch mode is:

- SCOP++ is started as background process and does not need any interaction with the user. [1]
- The commands are read from a procedure on a cmF-file.
- After termination of all commands (and sub-processes), SCOP++ shuts down.

6.4.1. Starting with an existing project

The following steps are necessary to run SCOP++ in command mode (assuming that the project already exists, e. g. with the name *myPrj*):

1. Write a cmF-procedure called "AutoStart" (case sensitive!). Note, that the default project's cmF-file (e. g. the file `exampleBatch.cmF` in the main project's directory) is written automatically when a new project is created (with an empty template of the procedure "AutoStart"). For details on how to write cmF-procedures, please refer to section 7.
2. Open a command window of your operating system. Change to the `bin`-directory of SCOP++ (if it is not included in your `PATH`-variable), e. g. to

   ```
   C:\Program files\inpho\scop++\bin
   ```

3. Run SCOP++ with the command line parameter `-M` followed by the keyword `BATCH` and a second parameter `-P` (specifying the project), e. g.:

   ```
   scop++ -MBATCH -PmyPrj
   ```

   If another than the procedure "AutoStart" shall be run, the name of the procedure can be specified by the command line parameter `-@`, e. g.

   ```
   scop++ -MBATCH -PmyPrj -@proc1
   ```

   If another than the default cmF-file shall be used, the name of the cmF-file (with its entire path) can be specified in brackets behind the procedure name, e. g.

   ```
   scop++ -MBATCH -PmyPrj -@proc1(C:\cmF_files\x.cmF)
   ```

[1] Unfortunately, the GUI of SCOP++ does not allow to run fully in background. Therefore, the main window of SCOP++ will appear also in batch mode. All further windows which are opened by the cmF-procedure will not appear on screen (only in the task bar).

If the file name includes blanks, the entire command line parameter must be quoted, with or without the option letter e. g.

```
scop++ -MBATCH -PmyPrj -@"proc1(C:\cmF files\x.cmF)"
scop++ -MBATCH -PmyPrj "-@proc1(C:\cmF files\x.cmF"
```

6.4.2. Starting with a new project

To start with a new project in batch mode is possible too. If the project specified by prameter -P doesn't exist it will be created automatically. In this case you have two options to run a command:

1. Create the project directory `myPrj` beforehand and within this directory the commandfile `myPrj.cmF`.
2. Specify the procedure to be used and the name of the cmF-file like with example above.

The remaining procedure is the same than with an existing project.

Note, that in batch mode, all questions to the user (and all warnings and error messages) are replied (accepted) by the default button.

6.5. Archiving Projects

Due to the use of a database, we recommend some additional actions when archiving projects (see section 9.2.7).

7. Command Language (cmL/cmF)

cmL/cmF is a special feature of the GUI of SCOP++. *cmL* stands for *command Line*, and *cmF* for *command File*. First steps in applying cmL were explained in section 4.6.

cmL and cmF allow for controlling the graphical user interface – and thus the processing altogether – by typing alphanumeric commands, and by calling procedures containing such commands. cmF is the means to control processing when SCOP++ is performed in the background, without displaying the graphical user interface. This capability is essential for assembly line type processing of data (such as regular country-wide mapping), but also for repeating complex proven command sequences.

For successfully applying cmL/cmF, the user needs to know more about specifics of the user interface and also of application architecture than for interactive work in ways familiar under graphical user interfaces such as Windows. Please refer to the appendix A for an introduction to the design of the program and cmL/cmF.

Consider this. Controlling a program by GUI actions involves just a few simple actions: the user can click at some spot on the screen with the left or the right mouse button; he can type alphanumeric characters: numbers, strings, and different signs. With cmL-commands (in a defined command syntax) the same actions can be done from only the keyboard. With cmF these commands are read from file.

7.1. The UserID and the Scope of UserID Interpretation

Control objects on the GUI can often be addressed from the keyboard by typing corresponding hotkeys; cmL addresses GUI objects via their names in a similar manner. These names are referred to as “userIDs”: they allow for identifying the corresponding object of the GUI. UserIDs are, though with some rare exceptions, immediately visible to users: they are written onto the buttons, displayed as labels to entry fields, to check boxes or radio buttons, represented in drop down menus.

The cmL-interpreter has to identify the GUI element by its userID. This identification is done in the scope of one window. Therefore the user has to declare the window whose GUI elements he wants to access as current window. This is accomplished by traversing the window hierarchy down to the desired window (”sublevel”).

Indirectly, the above means that windows are arranged as the branches of a tree, or rather as a directory tree. In applying cmL, the user has to trace this tree always starting from the main window. Any window (”sublevel”) entered via cmL becomes the current one; and all GUI elements belonging to the current window can now be addressed.

As the GUI of SCOP++ is a dynamic one, some GUI objects only become available in the process of working with a project, e. g. an overlay must be added to the project before its button can be addressed by cmL-commands. Furthermore, the names of some

userIDs are chosen by the operator while working with a project. For these two reasons, cmL-commands valid in a specific project might be invalid in another project. As a consequence we recommend to use the same names of overlays in various projects for which the same cmF-procedures are to be used.

Sometimes, GUI objects do not have a label (its userID as title) beneath them. A tool for finding the userID can be started by the drop-down element *Show cmL address* in the drop-down menu <*cmL*> on the main window of the application. Checking it (by clicking at it) turns on a special mode of showing "help tips"[1] appearing on the screen when the mouse cursor points to a GUI control element (drop down menu objects excluded), and stands still there for about a second: the help tips in this mode show the cmL addresses of the element. This can show the userID of a GUI element not visible on the screen as well as the hierarchy of parents. However, not all elements are accessible this way; in such cases, the help tip will ask you to try specification chaining (cf. 7.2.6).

7.2. The Syntax of cmL

Before going into details concerning cmL/cmF, it is inevitable to have some experience in working interactively with the user interface as presented by SCOP++: cmL/cmF is to a considerable extent mimicking the interactive actions; there is no table of commands or the similar: cmL/cmF applies the userIDs as seen on the GUI.

To start experimenting with cmL, open first the cmL communication by clicking at the drop down item <*cmL*> in the drop down menu of the main window. A pop-up menu appears, with the first element *cmL* on it; clicking at it opens a narrow window just high enough to carry a single field for entering strings: the cmL window with the cmL field – the field to type cmL commands into.

The pop-up menu <*cmL*> contains a further item *cmL Tutor*. Clicking at it opens a small tree of windows (subLevels). The cmL examples described in this and the following sections are based on these windows.

7.2.1. Accessing UserIDs

Let's assume that on the main window of our application there is a button, with the label written onto it *Model 1998 ...*. Left-clicking this button with the mouse, a window will be displayed (referred to as the "sublevel" to the button; triple periods ("...") at the end of userIDs indicate the presence of a "sublevel" – of a window, a dialog box, a file browser and the similar).

To mimic left-clicking a button in cmL, the userID is typed into the cmL field followed by a comma.

```
model1998,
```

As this example is situated under the drop down dialog *cmL Tutor*, you will have to type:

```
cmLTutor, model1998,
```

[1] Under Windows, "help tips" are referred to as "Quick Info", and their appearance can be customized by users.

and hit Enter to execute the command. After reading section 7.2.2 it will become clear why.

The rules applied here are:

1. Typing userIDs is not case sensitive.
2. The blank(s) in userIDs *must* be left out.
3. Typing the closing triple of periods is allowed but superfluous.
4. The userID is always followed by some delimiter (in this case a comma: ',').
5. No cmL line may end without some specific delimiter (forgetting the comma in this example results in an error message).
6. The cmL line is read by the system only on hitting the keyboard Enter key.

The system identifies the button *Model 1998 ...* based on its userID. Delimiter ',' is interpreted here as "select", corresponding to left-clicking the button. It is certainly easy to anticipate the importance of delimiters in cmL: typing the userID, one can identify some GUI object; the delimiter following it describes the action to be taken. Such actions are entering or opening-and-entering a window; returning from a window; opening the value field of the GUI object as identified – and closing such fields; but also less obvious actions such as "value chaining".

In case the system notices any error in the line as typed, it inserts an asterisk before the error, sends an error message, and waits for any next entries.

Rather than typing "model1998," one could also type

```
cmLTutor, Md98,
```

When typing userIDs, abbreviating them is allowed: the first letter must be there, and the rest must provide for unique identification within the realm of the GUI to be searched: within the current window (including all items of selections, cf. 7.2.5, and for the main window: including all items in drop downs – hidden pitfalls when entering shorthand forms of userIDs).

7. UserIDs may be abbreviated:
   - the first letter must be there;
   - the abbreviation must uniquely identify the userID;
   - letters closer to the beginning of the full word get higher weight (e. g. imagine two userIDs *mod12* and *mod21*: `mod1` will be identified as *mod12*);
   - and specifically, a short name is considered as identified if typed in fully, even if the userID thus typed could be an abbreviation for another userID in the same window (leaving out blanks and any ending periods).

7.2.2. Opening/Closing Windows, Entering/Quitting Sublevels

The term "window" is GUI specific. The corresponding term "sublevel" (covering more than just windows) is reflecting to the tree-type structure of windows: all windows have a "parent" (a button, a drop down item, and some others).

As we already know, for accessing a GUI object on a sublevel by cmL, it is necessary to trace the tree of sublevels and enter one sublevel after the other. This is done by successively accessing all ”parents” (being some buttons or some drop down items) – one after the other in the order of the window hierarchy – followed by special delimiters telling the system whether to open the window or just enter the sublevel without opening the window (i. e. not at all displaying it or just leaving it in the background).

If a userID is addressed via cmL, the following delimiter decides further interpretation. If the GUI object thus made *current* is capable of opening (e.g. a state-switch, cf. 7.2.3), and the separator given after the userID is a comma or a slash, then the sublevel "beneath" the object is ”entered” and becomes the current level for further searching.

Furthermore, in the above example, the window beneath the button *Model 1998 ...* is opened after interpreting the terminating delimiter ','; the same as if the user clicked at the button with the left mouse button. Note, that the sublevels to specific buttons (such as state-switches, cf. 5.1.1 and the following section) are not immediately opened by a left click. Thus they are not opened on a ',', neither.

A delimiter which will always open the sublevel is the '\' corresponding to a right-click at the button:

```
cmLTutor, Md98/
```

Both delimiters ',' and '/' enter the sublevel for interpreting following userIDs (i. e. they set the level for searching userIDs to the sublevel of button *Model 1998 ...*).

For the opposite:

```
cmLTutor, Md98\
```

closes the sublevel corresponding to closing the window by means of the operating system (e. g. clicking at the 'x' in the top right of the window or double-clicking the top left symbol).

When working interactively on the GUI, windows are usually closed by selecting a button such as *Close*, *OK*, or *Cancel*. The same is true for cmL: accessing one of these buttons via its userID will make the system close the window.

The delimiters '\' and ';' quit the sublevel for interpreting userIDs and return one level higher.

A liberal rule applies for drop-down menus: They may be seen as cascade of sublevels which may be entered successively, e. g.

```
cmL, cmLTutor/
```

or all items of drop-down menus may be seen belonging to the main window, which allows for accessing them directly:

```
cmLTutor/
```

Summarizing the rules:

8. Interpreting a cmL command always starts at the main window. Thus, the first userID specified must be one of the main window, including all drop-down items).
9. Remember the pair of delimiters '/' and '\': the first is to open the sublevel “underneath”, and the second is to close it.

10. The delimiters ',' and '/' enter a sublevel for userID interpretation, whereas the delimiters ';' and '\' quit it.
11. The delimiter ',' sometimes opens the window.
12. Accessing a button such as *OK*, or *Close*, or *Cancel* closes the window.

7.2.3. State-Switches

State-switches are non-standard buttons with a light on it (cf. 5.1.1, e. g. the state-switch *Isolines ...*). When interactively controlling the GUI, left-clicking the state-switch displays a pop-up menu to it with different entries – consisting, in a standard case, of two or more of the series *Properties ...*, *off*, *display*, *freeze*, *edit*.

Left-clicking the item *Properties ...* opens the sublevel – the window – "beneath" the state-switch. The other items represent different "states": left-clicking any of them changes the color of the light on the state-switch, indicating the state it has been switched to. The states *off* through *edit* are mutually exclusive: applying standard Windows GUI means, this functionality could be realized by four radio buttons with the corresponding names (userIDs).

Right-clicking the state-switch is identical to left-clicking the *Properties ...* item in its pop-up.

There are several different ways to mimic these actions in cmL/cmF. Inspect these examples:

```
cmLTutor, Md98, iso/
```

opens the sublevel first to button *Model 1998 ...*, and then to state-switch *Isolines ...*.[2]

```
cmLTutor, Md98/ iso,
```

will not open the sublevel to the state-switch *Isolines ...*: applying the comma as delimiter makes this to a "select" action, corresponding to left-clicking the switch. In contrast to the interactive action however, under cmL no pop-up menu appears.

```
cmLTutor, Md98/ iso display,
```

switches the state of state-switch *Isolines ...* to *display*, apparent by changing the color of the light correspondingly. The same action can be achieved by a couple of alternative forms:

```
cmLTutor, Md98/ iso display,
cmLTutor, Md98/ iso ( display ),

cmLTutor, Md98/ iso = display,
cmLTutor, Md98/ iso = ( display ),

cmLTutor, Md98/ iso disp,
cmLTutor, Md98/ iso d,
cmLTutor, Md98/ iso 1,
```

[2] If a window is opened, always the entire hierarchy of parent windows are opened, too, independent of delimiters after the parent's userIDs. Similarly, if a window is closed, all sublevels are closed, too.

The last three forms can also be written with the delimiter combinations of the preceding forms.

The corresponding rules:

13. The states of state-switches under cmL can be specified as:
    - written in full (in the standard case: `off`, `display`, `freeze`, `edit`);
    - at least the 1^{st} 4 characters of the state names entered without omission, or
    - just the first letter of the state names (in the standard case: `o, d, f, e`);
    - `0, 1, 2, 3`; to get the numbers, count the states allowed for the switch starting with 0.
14. The userID of the stateSwitch may be followed by different delimiters for these specifications:
    - blank or '=': in this case, it is only allowed to set a state; the sublevel won't be entered;
    - comma: in that case, the sublevel is entered. Nevertheless, the state of the parent state-switch may still be set.

7.2.4. Parameter Specifications

Windows can be considered as dialogs to specify "properties" of the parent (button, state-switch, drop-down item or the similar); and different individual properties can be considered as parameters, with attributes to them. Numeric and string type attributes typed in by the user are termed under cmL as "values". In this sense, specifications for a sublevel consist of parameter specifications, and the latter consists of userIDs and entries typed into corresponding fields accepting numeric or string type values. In cmL, groups of entries belonging to some parameter are separated from the next similar group by the delimiter comma.

Specifying values is the main subject of this paragraph.

Let's assume that on the sublevel to state-switch *Isolines ...* there are placed:

- a field for numerical entry labelled by the userID *Interval*,
- a field for alphanumeric (string) entry with the userID *Title*,
- a checkbox with the userID *Indexed*, and

The complex entry

```
cmLTutor, M98/ iso/ intval 5, tit=( The Vienna Woods ),
disp, Indexed "yes";
```

results in:

- opening the windows of the sublevel tree until the properties' window of state-switch *Isolines ...*,
- setting the value in the numeric field *Interval* to 5,
- setting the string in the text field *Title* to "The Vienna Woods",
- setting the state of the state-switch *Isolines* to *display*,

- checking the checkbox *Indexed*, and
- returning one level higher for userID interpretation, i. e. to the level of the parent element (in this example the window *cmL Tutor*). One could go on with specifications on that level, entering parallel sublevels, and so on.

The rules illustrated by this example:

15. The userID of a field for numerical entries followed by a blank as delimiter and a number will set the value in the numerical field to the number.
16. Closing the specification of parameters by the delimiter comma allows for specifying further parameters.
17. Parameter specification is in general allowed after a blank, after a '=', in parentheses, or by a combination of these.
18. Numerical values must be written with a dot as decimal symbol and without any digit grouping. Large numbers may be given in exponential syntax. Allowed are: `1.000` for 1, `1.00e3` for 1000, or `1E-2` for 0.01. Not allowed are e. g. `1,000.00` or `100 000`. Note, that this may be in contradiction to the appearance of numbers on your GUI.[3]
19. Alphanumeric values (strings) always have to be enclosed in parentheses.
20. Any type of attributes (both predefined and of type value) can be enclosed in parentheses; in this case, the parentheses are considered to be part of the token – i.e. `5` and `(5)`, or `d` and `(d)` are forms considered to be 'value'. This rule is also valid when the attributes are parts of specification chaining (cf 7.2.6).
21. Any parameter value may be quoted. If the value contains characters reserved for cmL interpretation, the quoted syntax is obligatory (e. g. `title( "The Vienna Woods (1:5000)" )`).
22. When following the userID of a checkbox, blank(s) as delimiter and reserved words such as `yes` set the checking state of the box. If no such reserved word is given, the state is toggled (as if the user clicked at it).
23. cmL automatically relates state specifications such as `disp` to the parent state-switch.

Some more examples:

Alternative syntactic forms for entering numerical parameters:

```
Intval 5,
Intval=5,
Intval = 5,
Intval ( 5 ),
Intval =( 5 ),
Intval ("5"),
Intval = ( "5" ),
```
etc.

The same for entering strings:

[3] In Windows, the appearance of numbers depends on the regional options of your operating system.

```
Title( The Vienna Woods ),
Title=(The Vienna Woods),
Title ( "The Vienna Woods ( 1:5000 )" ),
```
etc.

Some alternative forms for checkbox entries:

```
Indexed yes,
Indexed y,
Indexed 1,
Indexed ( y ),
Indexed(1),
Indexed = yes,
Indexed = ( yes ),
```
etc.,
the same forms for no, n, and 0.

One important rule concerning parameter specification was not yet mentioned:

24. In all cases, where in working interactively with the GUI left-clicking some button (e.g. the button *OK* or *Apply* for accepting parameter values) is needed, specifying it under cmL/cmF is also needed.

7.2.5. Selections

Selections are lists of similar GUI elements grouped into one element. The can have the type ”exclusive OR” (”XOR”, represented as combo-box or as group of radio buttons) providing a list of userIDs which can be chosen mutually exclusively. The second type are multiple-choice selections (referred to often as “OR-type selections”, represented as e. g. group of checkboxes) providing a list of userIDs of which the user can choose one, some or all.

The cmL syntax rules are valid for selections, too. Nevertheless, some specific remarks can be added:

25. Each element of a selection has its own userID and can be addressed by cmL directly. Entering the userID of the selection is allowed, but superfluous.
26. Specifying the userID of a selection element without explicate state specification to it (such as `yes`, `y`, `1` etc.) toggles the state of the corresponding member; with state specification, the state of the member will be set correspondingly.

    Exceptions for a XOR-type selection:

    - There is always exactly one element selected. It must not be de-selected. Therefore, the state of the *selected* element is not toggled when addressing its userID by cmL. The keyword `no` is not allowed.
    - Accessing any of the *non-selected* elements will automatically set the other elements to ”no”.
27. Extended syntax for OR-tpye selections: accessing the name (the userID) of the entire selection and, enclosed in parentheses, a comma-separated list of userIDs of those elements of it to be set as selected (i.e. to be set to “yes”). In this case, those members not listed are all set to “no” automatically.

7.2.6. Specification Chaining

Most GUI objects are labelled (they have a title, i. e. their userIDs) – but there are exceptions of importance also; two examples of such cases:

- a series of three numerical fields after the userID *Coordinates*;
- a series of different controls to allow for specifying an SQL select statement after the userID *Z-Accuracy*: a selection (combo-box, exclusive OR) with two userIDs *better than* and *worse than*, a numerical entry field, a selection with the two userIDs *AND* and *OR*, again a selection with two userIDs identical to those of the previous one: *better than* and *worse than*, followed by another numerical entry field.

The best solution for applying cmL in such cases is provided by "attribute specification chaining":

```
cmLTutor, chaing/ coords 1000.35 900.41 560.3,
Zaccuracy betterThan 5.0 and worseThan 2.0;
```

This illustrates the rules for "chaining":

28. A specification chain starts with a userID unique within the current sublevel (e. g. `coords` or `Zaccuracy`);
29. When applying blank(s) as separators, specifications following in a series are assigned to the GUI elements to follow one by one;
30. A specification chain ends at a non-blank separator. – Please remember, that in entering strings, they have to be written in parentheses; these parentheses are not considered as separators in the sense of chaining.

We recommend always to verify whether chaining is interpreted correctly, because the order of the GUI objects in the chain is not always intuitive. In cases where chaining is inevitable, however, the order in the chain is usually as expected.

There is another cmL functionality that could be helpful in such cases: to switch on the drop-down element *Show cmL address* (cf. 7.1)

7.2.7. Special Syntax Rules

31. File Browse symbols cannot be controlled by cmL (they are system services).
32. Characters following a double-slash ("//") are considered as comment, and will not be interpreted.
33. If a delimiter ';' closes the main (root) level, all characters to follow it on the same line will be ignored.
34. The maximum length of a cmL line is 1024 characters.

7.2.8. Dubious Cases

Think of a state-switch *Dubious* with the standard states *off*, *display*, *freeze*, and *edit*; and with a sublevel to it carrying among others two GUI objects with the userIDs *Offline* and *Disposition*. To address the latter, cmL entries with unintended effects could be:

```
cmLTutor, Dubious, offline, d;
cmLTutor, Dubious, offline, disp;
```

In both cases, the system will

- enter the sublevel of button *Dubious*,
- check whether `offline` could represent a valid specification for the state of the parent switch (it cannot),
- identify button *Offline*,
- check whether `d` (or `disp`) could be a state specification; the answer being affirmative, the state of state-switch *Dubious* will be set to *display* – which is most probably not the effect intended by the user.

The following forms yield the intended result ('selecting' the button *Disposition*):

```
cmLTutor, Dubious, offline, dsp;
cmLTutor, Dubious, offline, dispos;
cmLTutor, Dubious, dsp;
```

Examples with the syntax of value specification, yielding unintended results:

```
cmLTutor, Dubious dsp;
cmLTutor, Dubious(dsp);
cmLTutor, Dubious = ( dsp );
```

As there is no delimiter ',' or '/' after the userID *Dubious*, the sublevel will not be entered. In these cases, `dsp` must be a specification of the state of the switch; however, `dsp` is no proper abbreviation for *display*: an error message results, telling the user that the state of "Dubious" cannot be specified as "dsp". The same would happen if specifying `disposition` rather than `dsp`. (Remember that in specifying states, the abbreviation may be just the single first letter of the name (for *display* `d` or `D`), or else it must carry at least the first 4 letters of it; i.e. `disp`, or for instance `dispy`, but not `di`, or `dsp`).

The rationale for

- first checking the entries against possible states of the parent state-switch, and only afterwards against userIDs belonging to the sublevel, and for
- the restrictions in abbreviating state names

is a reduced probability of unintended effects: on the one hand, it is easy to overview the names of the possible states, but the same may be confusing with regard to numerous and sometimes hidden userIDs belonging to the sublevel. Applying stricter rules of abbreviation to these names helps avoiding unintended identification also.

Now consider the case illustrated by some further GUI objects on the sublevel to state-switch *Dubious*: let it be a numerical entry field with the userID *numVal*, and a state-switch with the userID *Switch*. Some dubious cases of cmL entries:

```
cmLTutor, Dubious, numVal 2 3;
```

The numeric field *numVal* is specified to become 2. This specification is followed by a blank as delimiter: the entry `3` will be chained to the state-switch *Switch*; as a state specification, it corresponds to *edit*. The light of the stateSwitch will change accordingly to red.

It is doubtful, whether the user really intended what happened. If he did, he should have rather typed

```
cmLTutor, Dubious, numVal 2, switch 3;   // or edit or just e
cmLTutor, Dubious, switch e, numVal 2;   // alternative
```

A comma past the first value:

```
cmLTutor, Dubious, numVal 2, 3;
```

results in an error message complaining about a dangling numerical entry. Typing here an `e` rather than `3` could either identify some unintended GUI object with a userID started by an *e* (provided that this abbreviation is unique), or in an error message telling that the word ”e” cannot be identified.

7.3. Summary of the Main Rules Using Delimiters

The main delimiters are the comma, the blank, and the semicolon:

- commas separate groups of specifications for different parameters including their attributes (whether pre-defined or of type value). Any userID past the comma is searched for on the entire sublevel;
- blanks are in the most common sense the delimiters providing for chaining attribute specifications. In this sense, entering a userID, a blank and a value can also be considered as simple chaining;
- semicolons close specifications for the current sublevel. After encountering a semicolon, the sublevel carrying the parent object becomes the current one;

And:

- string type values must be enclosed in parentheses.

7.4. The cmL Buffer

The cmL window stores recent cmL commands for retrieving them. If the command line is interpreted successfully (but not otherwise), then it will be appended to a list of such lines: to the “cmL buffer” (to carry the most recent 32 lines). Using the keyboard arrows, the user can recall any previous line for editing and/or re-entering it. If a re-entered line has not been edited, it is not going to be inserted into the list a second time; in this case however, the pointer to the list becomes the line’s original position on it: this allows for re-entering one-by-one a previously formulated series of commands. The userID as identified by the first *word* on the cmL is used as an identification of the line: entered past a ‘!’ character (short hands are sufficient), the line is searched for on the list, and displayed on the cmL for editing and/or re-entering. Entering just "!;" moves the list-pointer to its most recent position.

7.5. Writing cmF Procedures

Command procedures are command lines (cmL) within the frame of a `@procname;` and of an `@endProc;` statement, where *procname* may be any name given to the procedure.

Both these statements must be placed onto individual lines with '@' as the first non-white character on them, and closed by semicolon.

The body of the procedures consists of any number of command lines as described earlier in this section – with one important difference: on command files, entering new lines (CR and/or LF) will NOT reset the level of interpretation to the main window.

Each line on a command file (cmF) must end with some proper delimiter (this is just as true of cmL, naturally). – This provides for writing procedures in legible form enhanced by indentation to reflect window levels and other structural specifics.

Calling a procedure is performed by entering the `@procname;` statement on the command line facility cmL as provided on the main window of the application. Optionally, the name of the command file may also be specified as parameter to the `@procname`, enclosed in parentheses – e. g. `@myproc(myfile.cmF);`. The command file entry is stored for further calls as the current command file; the first such entry overwrites the default command file name as specified or constructed at program start (application dependent).

Procedures may call each other. For this, a statement in form `@procname;` is used, relating to a procedure on this same command file; at this point, procedures on other command files cannot be addressed.

The statements `@procname;`, `@endProc;`, and `@procname;` may not be followed by further statements on the same command line.

For writing cmF-procedures, the cmL syntax rules are fully valid. Following additional rules apply:

1. Empty or blank lines are allowed.
2. Lines consisting of just a comment (after '//') are allowed.
3. Except for in special cases, the first specification encountered on any line must be a *word* (the shorthand of a userID) followed by at least one *delimiter*; this userID is always searched for on the main window.
4. The userID of any GUI object belonging to the current window can be entered directly. The GUI object identified this way is made the *current* one.
5. In addition to the single-line syntax in parentheses, there is a special syntax provided for entering text containing multiple lines:

   ```
   userID(
   Line 1
   Line 2
   ...
   Last line);
   ```

   The closing parenthesis ')' may be written in the next line also. Furthermore, so to allow for including special characters in the text, the opening '(' can be followed by a quote character (with any blanks before it ignored); the closing ')' must be preceded by the corresponding pair. The following pairs are supported:

   ```
    (" "),   (' '),   (* *),   (/ /),   (\ \),
   (< >),   ([ ]),   ({ }).
   ```

For reading how to start SCOP++ in *batch mode*, please refer to section 6.4.

7.6. A Command File Example (cmF)

```
// ================================ CallAll
@CallAll;
   @Model;
   @Chaining;
   @Dubious;
@endProc;

// ================================ Model 1998
@Model;
   cTutor/
      Mod98/
         Iso, f;
   @Iso;
@endProc;

@Iso;
   cTutor/
      Mod98,
         iso/
            int 5,
            indx yes,
            title( Vienna Woods ),
            displ;
@endProc;

// ================================ Chaining
@Chaining;
   cTutor/
      Chain/
         coords 4444.44 5555.55 666.66,
         Zacc worse 10 or better 1;
@endProc;

// ================================ Dubious
@Dubious;
   cTutor/
      Dub/
         offl,
         dsp,
         fr,
         ed,
         num 2,
         sw f,
         display;
@endProc;
```

7.7. Table of Syntax Rules

Word, name	Description
UserID	shorthand, non-case-sensitive. Identifies an uxObject as current
".."	reserved *word*: synonym to *separator* ';'
@procname	procedure to be searched for on the current commandFile (cmF); case sensitive

Separator	Description
blank | =	specifications to follow; blank: chaining
,	(comma) – the next word on the line is searched for on the entire *current* level.
/	*word* must be the userID of an object capable of **open**ing. Action similar to ',' but the subLevel is going to be displayed on the screen
;	End of specifications for a sublevel (window); move one level up
\	as ';' but the sublevel, if displayed, will be removed from the screen
(	see table below for *specification*s
> <	**stateSwitch**es only: raise/decrease state; the number of these *separators* in a series defines the extent of this action (e.g. >> corresponds to raising the state by 2)

Specification	Description
number	interpreted as value for the object identified by *word*
(	must be followed by one of the following, and closed by a parenthesis: • a number for numerical fields, o/d/f/e or other forms for stateSwitches, y/n or 1/0 for ddItemChecks and OnOffSwitches • a string or text (with a special syntax) for string type fields, • (list of) name(s) for complex GUI objects and areas: for "Selections"; in this special case, names not listed will be de-selected[4]. • a file specification for @procname. This specification redefines the current command file
o | d | f | e y | n 0 | 1 | 2 | 3	1^{st} letter of reserved names, or the corresponding number started with 0 stateSwitches: Off / Display / Freeze / Edit check items: yes / no

[4] This is only true for specifying the list in parentheses; without parentheses, the state of them will be toggled as if selected by a mouse click or via hotKey.

Part IV.

Reference Manual

8. Working with Graphics

Within SCOP++ there are several windows showing graphics data. The most prominent of them, the main graphics panel constitutes the central part of the main SCOP++ window. All of these graphics windows share some common features which will be described in this chapter. Figure 8.1 shows a graphics window including all of the potentially available elements. The main building blocks are the *graphics area* in the middle of the window, a horizontal toolbar with *general tools* at the upper border, a vertical toolbar with *editing tools* at the left side and a status bar with *coordinate display* at the bottom. Some graphics windows may show only some of these elements or none of them. The following sections describe them all in detail.

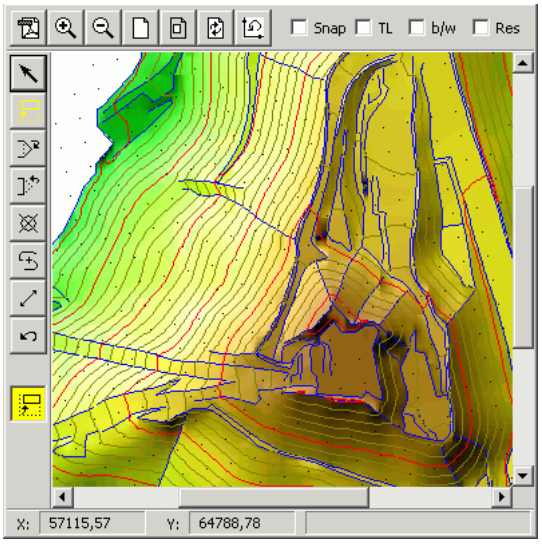


Figure 8.1.: A Graphics window

8.1. The Graphics Area

Two main types of graphics data can be distinguished:

Vector data such as data plots or contour maps,

Image data such as orthophotos, hill-shading, scanned maps, etc.

Vector data is always displayed in front of the image data. Multiple overlapping images will be mixed to a single one; a good example of this is displaying hill-shading and Z-coding concurrently.

8.2. General Tools

Above the graphics area there is a horizontal tool bar containing tools for manipulating the visual appearance of the data on screen (see Fig. 8.2).

Figure 8.2.: General tools of a graphics window

The following tools are available:

PDF Output: Output of the data on screen to a file in Portable Document Format (PDF) as defined by Adobe Corporation. A window will appear on screen where the following parameters may be selected:

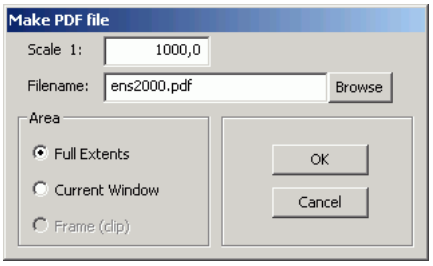


Figure 8.3.: Output graphics onto file in PDF format

- The numeric field *Scale* to specify the scale of the drawing,
- the text window *Filename:* to specify the output file for the drawing, and
- a group *Area* with 3 radio buttons to select the area to be written to the PDF file. With radio button *Full Extents* selected all the displayed data will be written, the data within the actual viewport of the graphics area may be selected with radio button *Current window*. The radio button *Frame (clip)* will be availabe only if the state-switch *Frame* is at state *display*. In this case the frame and the displayed data within the inner border of the frame will be written to the file. Clipping of the data takes place at the inner border of the frame.

This dialog window is not available in batch mode. In that case the drop-down menu *Project/Pdf output...* has to be used for PDF output.

Zoom Plus: Double the display-scale.

Zoom Minus: Reduce the display-scale by factor 2.

Zoom All: Show all data.

Zoom Window: Show a selected part of the data. The clipping rectangle can be selected with the mouse left-clicking at two diagonal corner points (no dragging). If one point has already been clicked at, you can interrupt the action with clicking the right mouse button. This function is active only temporarily and will be switched off after executing it once.

Redraw: This button allows manually triggering a redraw of the graphics area. It may be helpful in some situations when the display is corrupted in some way.

Rotate: This button allows to rotate the graphics area on screen. This may be helpful when using local coordinate systems not oriented north up, especially when using images.

Snap *Snap:* With this checkbox selected snapping to an existing point is possible when editing a graphics element or within the measuring function.

Thick Lines: Selecting this checkbox will add 1 pixel to the linewidth when drawing on screen.

Grayscale: With color images in the background the representation of vector data is often difficult to differentiate. Selecting this checkbox will switch the image representation on screen to grayscale.

Resample: When this checkbox is selected images will be drawn on screen with bilinear resampling giving a smoother appearance; otherwise, nearest neighbor resampling will be used.

8.3. Coordinate Display

Below the graphics area there is a status bar with two numeric fields continously displaying x and y coordinates when the user moves the mouse cursor over the graphics area. (see Fig. 8.4). The field at the right side is used to display a progress bar when loading data to the graphics panel.

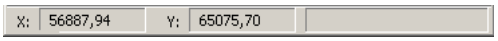


Figure 8.4.: Status Bar of a graphics window

8.4. Editing Tools

At the left side of the graphics area there are buttons to select editing functions. Within all functions explained below every button press makes one step in the editing process. It is never necessary to drag the mouse while holding down a button. Within most functions it is possible to confirm the action with the left mouse button or to cancel the function with the right mouse button.

> **For editing of data it is necessary to switch this data to state *edit* (see section 5.1.1).**

8.4.1. Select Graphics Element


With this function active, clicking at an element with the left mouse button selects this element. Usually this function will not be selected manually by the user but the program system switches to this function at several places when selection of an existing graphics element is needed, eg. when selecting radio button *Object* within group *Digitizing Mode* in the window *Limits*.

8.4.2. Modify Graphics Element


With this function active, clicking at an element with the left mouse button selects this element. It will appear in red color and the points will be marked with small rectangular boxes. The position of a point can now be changed by clicking at it. “Rubberbands” appear and the point can be dragged while showing the coordinates in the status

bar below the graphics area. A next click with the left mouse button will fix the point at the new location. Depressing the right mouse button cancels the operation. The element stays in selected state.

Clicking at a point of a selected element with the right mouse button opens a window, where coordinates of the point will be displayed. New coordinates can be typed in. *OK* confirms the changes, *Cancel* closes the window without changing the coordinates of the point.

Selecting the checkbox *Z(Poly)=const* causes a change of the z coordinate of all points from the selected polygon. If all points of a selected polygon have the same z coordinate (e.g. contour line), then this checkbox is preselected by default.

8.4.3. Insert a New Point

 Pressing the left mouse button while the mouse cursor touches the edge of a polygon initializes the insertion of a new point beetween the endpoints of this edge. "Rubberbands" appear which are moving with the mouse cursor. Pressing the left mouse button again inserts a new point with coordinates from the current location. Pressing the right mouse button cancels the insert action.

The z coordinate of the new point will linearly interpolated from the z coordinates of the neighboring points. If this is not appropriate then the z coordinate can be changed with the function *Modify Element*.

Clicking at the startpoint or endpoint of the polygon when selecting the polygon allows to insert a Point at the beginning or end of that Polygon. A single "rubberband" will appear, which is moving with the mouse cursor. Clicking again with the left mouse button at the appropriate position will add the new point at the beginning or end of the polygon. Pressing the right mouse button cancels the action.

The z coordinate of a new point at the beginning or end of a polygon will be extrapolated from the first or last edge of the polygon. If this is not appropriate then the z coordinate can be changed with the function *Modify Element*.

8.4.4. Delete a Point from a Polygon

 When clicking at a point of a polygon "rubberbands" appear showing the new geometry of the polygon after deleting this point. Clicking again with the left mouse button confirms the action, clicking the right mouse button cancels the operation.

8.4.5. Delete an Element

 Clicking at an element with the left mouse button selects this element, it will appear in red color and the points will be marked with small rectangular boxes. Clicking again with the left mouse button will delete the element, clicking the right mouse button cancels the operation.

8.4.6. Join Two Polygons

Two polygons have to be selected with the left mouse button. Clicking the left mouse button again will join the two polygons. The polygon selected last will be added to the polygon selected first. The polygons will be connected at the shortest possible gap, the directions of both polygons will adapted as needed. Clicking the right mouse button cancels the operation.

8.4.7. Cut a Polygon

A polygon has to be selected with the left mouse button. Clicking the left mouse button again while the mouse is located at a point of the selected polygon will cut the polygon in two parts. Clicking the right mouse button cancels the operation.

8.4.8. Measuring Function

After clicking at two points within the graphics area the following information window will appear on the screen:

Shown are the coordinates of the two points, the coordinate differences, the planar length with the xy plane and the 3d distance between the selected points.

8.5. UNDO-Function

Clicking at this button causes an undo of the last editing action.

The undo buffer is limited only by the available memory but it will be cleared every time when data is removed from the graphics panel (e.g. switching the display state from *edit* to *off*).

8.6. Current State

This button with yellow background shows the current active editing function (current state). Clicking at this button will turn off the current editing function.

9. The Menu Bar

Roughly this chapter deals with the GUI items which can be found in the menu bar of the main window. This comprises the creation and access of Scop++ projects (see section 9.2) and the manipulation (adding and removal) of the overlays (see section 9.3) of a project.

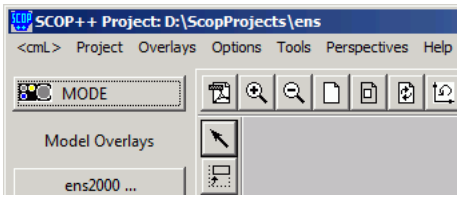
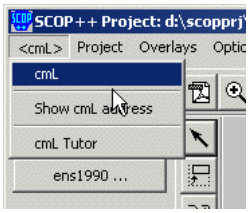


Figure 9.1.: The menu bar of the main window

9.1. Command Language



The bar item <*cmL*> allows for working with the command language *cmL* interactively (cf. 5.1.4 for an

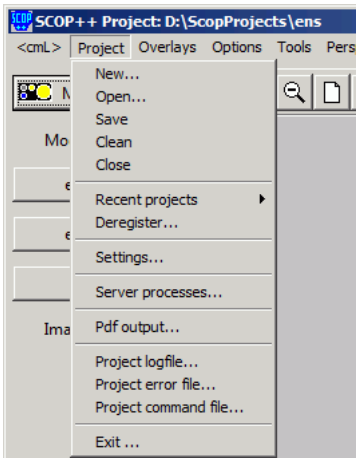
introduction and section 7 for a detailed description).

By selecting entry *cml* from the bar item <*cmL*> a narrow window is opened consisting of a string line for entering *cmL*-commands (this window can also be opened by the keyboard by typing <Alt+c> followed by <c>, and also by <Alt+c> followed by <Alt+c> again). For executing *cmL*commands, hit the <return> key on your keyboard. The arrow keys up and down can be used to scroll in the buffer of recent *cmL*-commands.

Checking the entry *Show cmL address* activates 'quick help tips' in a mode to display the *cmL*-address rather than the quick help string: when the mouse pauses for a while at some spot chosen by the user, 'quick help tips' will appear as little temporary yellow fields at the mouse point - provided that the option is active, and that a corresponding quick help string has been specified for the spot pointed at. In the current version of Scop++, quick help tips are at most spots inactive; with the entry *Show cmL address* checked however, the option will be set active, and the *cmL*-address will be displayed including its entire scope of hierarchical windows. This might help users in writing *cmL*-commands.

Unchecking entry *Show cmL address* will restore the original mode of displaying quick help tips.

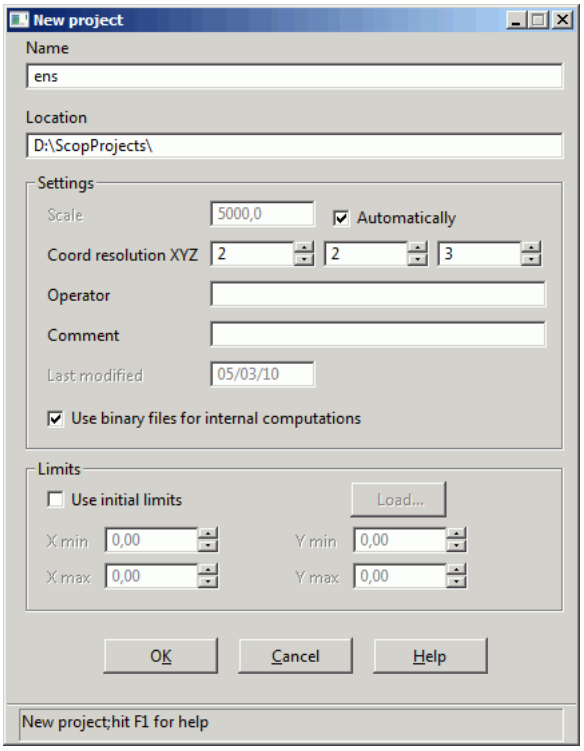
9.2. Project Manipulations



The widgets concerning this section will be found in the menu bar of the main window beneath the bar item *Project*. They handle

- Creation of new projects
- Opening existing projects
- Saving the current project
- Cleaning the current project
- Closing the current project
- Adapting settings of the current project
- Viewing server processes
- Creation of a PDF file with contents of main window
- Viewing the logfile of the project
- Viewing the error file of the project
- Viewing/Editing of the command file
- Exiting SCOP++

9.2.1. New Project



To create a new SCOP++ project open the *Project/New...* dialog by selecting entry *New...* from within the bar item *Project*. A dialog window opens, where the name and the location of the new project can be specified. Type the name of the new project in the text field *Name*. The project name will be automatically appended to the path of the SCOP Projects directory specified at installation time. If you don't want to create your project at the default location, enter the path to the project directory in the text field *Location*. Although it is possible to create the project anywhere on the computers hard disk, it might by advantageous to use the default path. Pressing the button *OK* will create a directory with the correct name at the desired location. In this directory the project file (*.spr) and the project table (*.TOP) will be created. Mind that you can't manage two projects with the same

name in the database. Therefore the attempt to create a new project with the name of a project already in the database will produce an error. If there is already a project file in the chosen directory with the name you have typed in you will be asked if you really want to overwrite it. Accepting this the old project file will be overwritten, else no project will be created. A correct creation of the new project can only happen, if the old project isn't in the database anymore (see above). The project names are treated case-insensitive, although you should always see the name exactly as you typed it, but internally the names *Ens*, *ens* or *ENS* are the same. It's a strong recommendation to use lower-case project names, not to get confused.

Some additional settings can be made within this window. You can specify the general scale for the whole project in the numeric field *Scale* or let the scale be set automatically. If you want to set the scale manually, uncheck the checkbox *Automatically* besides the numeric field *Scale*. This will make the numeric field *Scale* accessible and the value can be entered. Please take care of a proper setting for the general scale the since numerous subsequent project settings are dependent of the scale (text height of annotations, ...). The automatic determination of the scale will select a value out of set of common scales (500, 1000, 5000, ...) such that the overall extensions of the project will fit to an A0 plot.

With numeric fields *Coordinate resolution XYZ* different resolutions for XY and Z may be chosen. This settings will be used for storing the data and internal computations and cannot be changed later on.

The text fields *Operator and Comment* are optional and evident.

Below the last one are the text field *Last modified* denotes the last modification time of the project file (and therefore also the project).

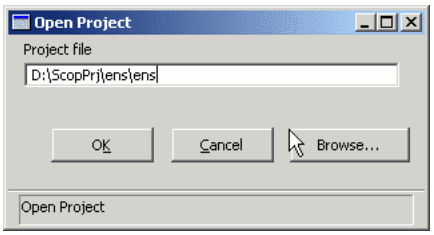
The checkbox *Use binary files for internal computations* is selected by default for efficiency of processing. If one like to inspect some of the internal used files using a text editor this option may be changed.

For some projects it may be convenient to set the limits of the project at this very first place. This can be done selecting the checkbox *Use initial limits* and entering the corresponding values to the numeric fields *X min, Y min, X max, Y max* or loading them from a file.

It goes without saying that an open project will be saved and closed before a new project will be created. After a new project has been created successfully, the title of the main window will be changed to reflect the project name. In reverse a close (see section 9.2.6) will reset the title.

In each project directory a project file named *ProjectName.spr* will be created. In this file some information concerning the projects are stored (e.g. names and types of all overlays, scale, limits, ...). In this way the application can restore the state in which the user has exited a project the last time.

9.2.2. Open Project



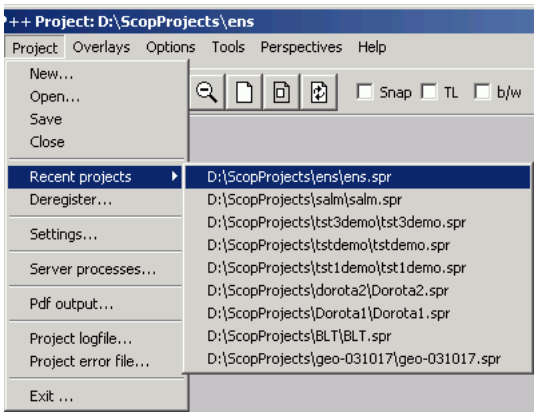
To open an existing project, select entry *Open...* from within the bar item *Project*. The text field *Project file* will accept following values:

- The project name only, if the project is in the default project path.
- The project directory.
- The path and name of the project.

You can also browse for the project file with a file browser opening if you click on the button *Browse...*. Select the desired project file in the usual way and click the button *Open* of the browser. After pressing the button *OK* the project will be opened. Alternatively you can also accept the typed in project name by terminating it with an carriage return (Enter).

As SCOP++ can hold only one single project open, the current project will be saved and closed correctly when another project is opened. After a project has be opened successfully, the title of the main window will be changed to reflect the project name.

9.2.3. Recent Projects



For a quicker access, the ten most recent projects can be opened

by the means of a recent files list. To open one of them select the entry *Recent projects* within the bar item *Project*. A list of the most recent projects will be displayed. Selecting one of the entries listed there will open this project. If the desired project is not in the list of the most recent projects, use *Project/Open...* to open the project as described above. The entry *Recent projects* won't be accessible if you run Scop++ for the first time, because there aren't any projects which have been opened. The files list will is sorted by the order of opening.

As SCOP++ can hold only one single project open, the current project will be saved and closed correctly when another project is opened. After a project has be opened successfully, the title of the main window will be changed to reflect the project name.

Deregistered projects (see section 9.2.7) will not be immediately removed from the recent file list. This makes it easy to insert them into the database again by only selecting the correspondent item in the recent file list.

9.2.4. Save Project

To save the current state of a project, select the entry *Save* from within the bar item *Project*. Doing so, the project files (*.spr) and the parameter files of all model and image overlays

(*.stp) will be saved to disk. There is no *Save As...* function offered here, because the saving of the parameters of overlays is seen as an export functionality and can be found in the appropriate section (see 12.2). In the ordinary workflow there should be no need for an explicit call of save, since the application takes care that all important data will be saved automatically. This function is simply for the user who wants an explicit save at certain moments.

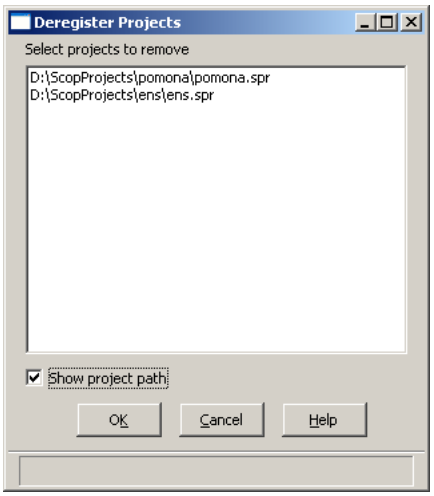
9.2.5. Clean Project

During the work on a project several intermediate files are produced by SCOP++. Among them are graphics files for contour lines, shadings, z-codings and many more. To delete all intermediate files select entry *Clean*.

9.2.6. Close Project

Selecting the entry *Close* from within the bar item *Project* will close the project. All the graphics displayed on the main graphics panel will disappear as well as all state-switches for the overlays. Closing a project explicitly is necessary if you want to deregister the current project.

9.2.7. Deregister



As mentioned in sections above, SCOP++ contains an internal database. For every new project an appropriate database table is created and data read from input files are stored in this table. When working with SCOP++ the number of projects may either grow large or a project is finished and needs to be archived. In both cases it is necessary to deregister the project. Deregistering a project means, that the project table is cancelled from the database catalogue. The project table itself (*.TOP) will not be deleted, since you might want to re-activate the project at a later time. To deregister a project select entry *Deregister...* from within the bar item *Project*. A dialog window will open, where the list of active projects is displayed. Selecting one or more projects from the selection *Select projects to remove* and clicking the button *Ok* will deregister the selected projects and close the window. To close the window without deregistering a project select the button *Cancel*.

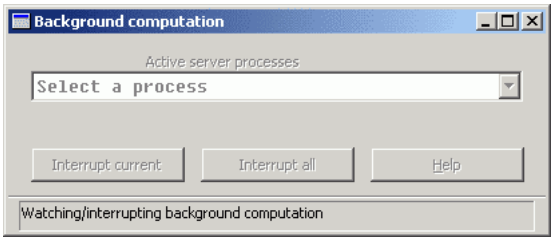
When a project is deregistered, it is ready to be archived, by just copying the whole directory tree to an archive (CD, ZIP, Tape, ...). In this state it is allowed to delete the project table file, holding the input data of the whole project. To re-activate a project, the directory tree of the projects -including the project file *.spr and the project table (*.TOP)- only needs to be copied from the archive to an appropriate place of the hard disk. The

project directory can be restored anywhere on the computers hard disk. Be aware, that the read-only flag is turned off, when restoring a project from the CD. A project can be re-activate just be opening it in the way described above.

9.2.8. Settings

The parameter settings of the project as defined during the creation of the project can be changed by means of the drop down dialog *Settings*. For further information concerning the meaning of the different parameters please refer to section 9.2.1.

9.2.9. Server Processes



Beneath this item an interrupt/progress bar window will open. If you open it without having started a server process all elements will be locked (see figure 9.2.9). It will be opened and closed automatically when an interruptable process is performed. In the selection *Active server processes* the process which shall be interrupted can be selected. In most cases this selection will only hold one element but as SCOP++ is a application with tasks running concurrently, there can be more than one process active. The selection will be automatically updated in a way that the process, whose progress is the most advanced will be put in first place. If there is more than one active process, all of them can be interrupted, by pressing the button *Interrupt all* and the current selected process, by pressing the button *Interrupt current*. If there can be made some statements of the progress this will be done by a progress bar on the bottom of this window. Also this progress bar indicates the progress of the selected process.

9.2.10. Pdf Output

This item opens a window to initiate a pdf output of the contents of the main graphics panel (see section 8). This window is the simplified form of the window beneath the button *Pdf* of the main graphics panel (see figure 8.3). It only allows specification of the output file (text field *Pdf file* and/or button *Browse*) and the scale (numeric field *Scale*). A the value of scale will be set to the value specified in the dialog window *Settings* or in the dialog window *New Project* (see section 9.2.1 and 9.2.8). This dialog serves for pdf-output in BATCH-mode, as in this mode the graphics area is not accessible.

9.2.11. Project Logfile

This item will open the selected editor (see 9.4.6) and display the logfile of the project. For determining the extent of protocol please refer to section 9.4.4.

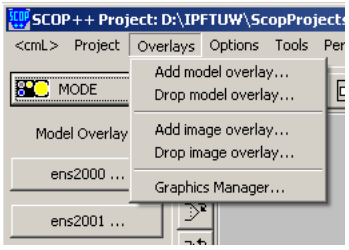
9.2.12. Project Error File

This item will open the selected editor (see 9.4.6) and display the error file of the project.

9.2.13. Exit

Selecting this item a save and close of the current project followed by an exit of SCOP++ will be performed.

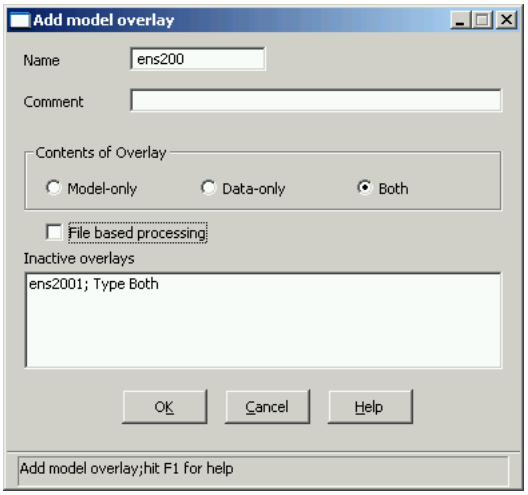
9.3. Overlays of a Project



The section covers the drop down dialogs below the bar item *Overlays* (see figure 9). The bar item *Overlays* will be locked as long as no project has been opened. The items of this bar item will be locked resp. unlocked logically. That is for example that the drop down dialog *Drop model overlay...* will be locked if there isn't a model overlay to drop.

In each overlay directory the application creates a setup file named *OverlayName.stp*. In this file the information concerning the overlay are stored. In this way the application not only can restore the state in which the user has exited a project the last time, but can also load parameter specifications from another project into the current one (see section 12.2).

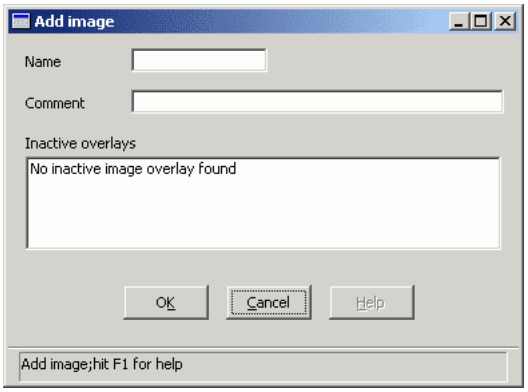
9.3.1. Add Model



This dialog serves for adding model overlays. It will be opened automatically if a new project was created or a project without an overlay was opened. In the text field *Name* you can enter the name of the model overlay. The name of the overlay equals per default the name of the subdirectory in which the primary data for the overlay is searched for. If you want to have your data elsewhere, you have to specify its location in a later step (see section 12.2.1). The second text field *Comment* can be used to provide a few words which will appear in the status field of the overlay window. The three radio buttons *Model-only, Data-only, Both* specify the content of the model overlay which will be inserted. For overlays of type *Both* the checkbox *File based processing* is accessible allowing to switch between file based or database processing mode (see section 5.3.11 for more details). Clicking on the button *OK* will insert the overlay. This implies the import of the data, if the data file is found (see above), the depiction of relevant data in the window *Limits* and the display of a button *Model Overlay*. Behind this button, situated on the left margin of the main window below the label*Model Overlays* the window *Model Overlay* will open.

The selection *Inactive overlays* serves for simplifying the task of re-adding overlays which have been removed. Its entries consist of the name and the type of the overlay. The contents is derived from *.stp files found in overlays directory of the current project, which aren't inserted as overlays yet. Simply select one of the items and the overlay will be inserted again. If there aren't all of the inactive overlays which should appear in the selection, please close and open the dialog which will update the selection *Inactive overlays*.

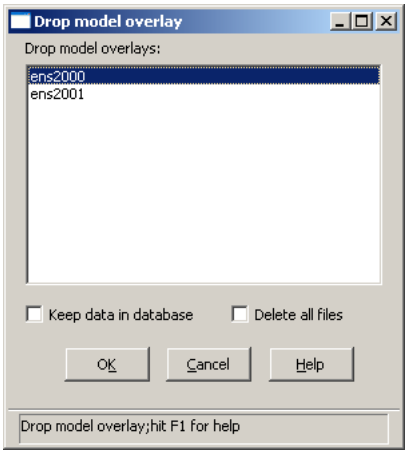
9.3.2. Add Image



This dialog serves for adding image overlays. As in the window *Add model* you can specify a name and a comment. The name of the overlay can be different from the name of the image data file. The image data file has to be imported after creating the overlay. Clicking the button *OK* will insert the image overlay. This implies only a display of the button *Image Overlay*. Behind this button, situated on the left margin of the main window below the label *Image Overlays* a window titled with the name of the overlay will open (see section 13.2).

The selection *Inactive overlays* servers for simplifying the task of re-adding overlays which have been removed. Its entries consist of the name and the type of the overlay. The contents is derived from *.stp files found in overlays directory of the current project, which aren't inserted as overlays yet. Simply select one of the items and the overlay will be inserted again. If there aren't all of the inactive overlays which should appear in the selection, please close and open the dialog which will update the selection *Inactive overlays*.

9.3.3. Drop Model



In the selection *Drop Model Overlay* you can select the model overlays which should be dropped. Clicking the button *OK* will remove the overlays from the current project. On response to this action the overlay buttons on left margin of the main window and all graphics of the respective overlays are removed from the screen. If the checkbox *Keep data in database* is not checked, then all the input data, which was imported to the projects, are also removed from the projects database table. Note that because of buffering the database table file *PROJECT.TOP* located in the projects root directory may not become smaller, although data is removed from the database.

If the checkbox *Keep data in database* is checked, then the respective data remains in the database. This is useful, if you want to re-add the model overlay to the project at a later time. This can be done by selecting the overlay from the list of inactive overlays from within the dialog window *Add model* (see section 9.3.1).

Note also that on *Drop model* no files will be deleted from the computers hard disc when the checkbox *Delete all files* is not checked. This includes also the setup file *overlay.stp* holding all the parameters. Otherwise (if the checkbox *Delete all files* is checked) all files and directories related to the selected overlay will be definitely deleted from the hard disc.

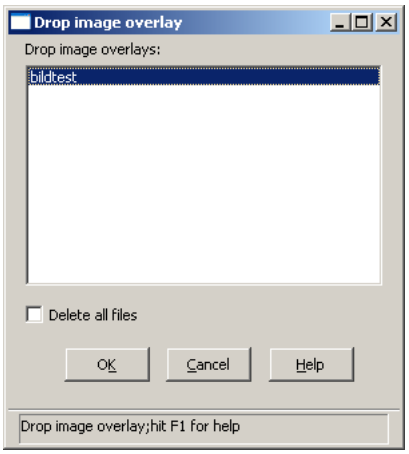
A commandline / commandfile example for this window is:

cmL example

```
@DropOverlays;
  DropM/,
    // Assuming there are overlays: e2005, e2006, e2007
    NameSel=(e2005,e2006,e2007),
    DeleteAllF=1,
    OK;
@endProc;
```

9.3.4. Drop Image

In the selection *Drop Image Overlay* you can select the image overlays which should be dropped. Clicking the button *OK* will remove the overlays, i.e. remove all graphics, buttons etc. from the screen. If the check-box *Delete all files* is checked, then the respective files and directories to the selected image overlays will be definitely deleted from the hard disc.



9.3.5. Graphics manager

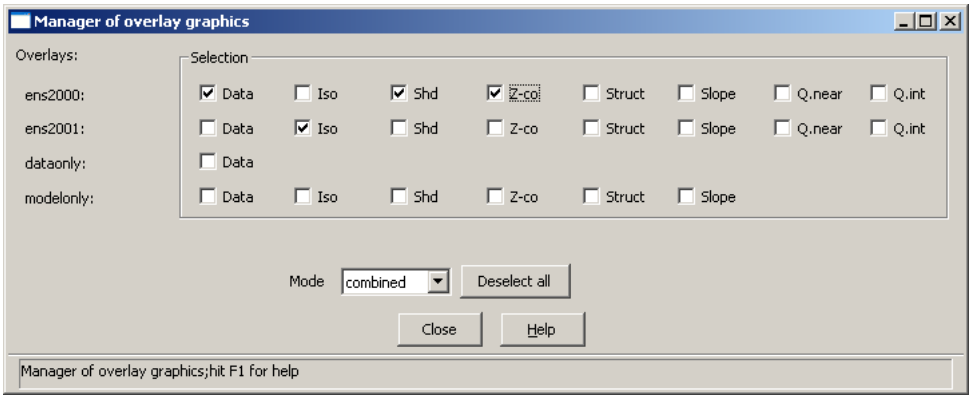


Figure 9.2.: Graphics Manager window

The dialog window *Graphics manger* provides an additional way to display DTM graphics like isolines, hill shadings, etc. Normally these graphics are activated using the appropriate buttons of the respective model window, which often requires a series of mouse clicks. The *Graphics manager* offers a simplified way. To activate a certain graphic (eg. isolines of model ens2000) just click the appropriate radio button or check box.

Graphics can be displayed in two different modes:

- combined: The switches are diagramed as check boxes indicating that multiple graphics can be selected simultaneously. As far as possible, all the activated products are overlaid to construct a single composed graphic.
- exclusive: The switches are diagramed as radio buttons indicating that only one graphics product can be selected at a time. This mode is above all useful to quickly compare similar graphics eg. shadings of different DTM version.

9.4. Options

In this drop-down menu you can define settings which will be valid for all projects. These comprise

- The management of feature codes (to be used in code conversion tables).
- The management of code conversion tables (controlling data import and export).
- Enabling/Disabling protocol output.
- Enabling/Disabling error protocol output.
- Enabling/Disabling interactive mode.
- Enabling/Disabling unattended mode.
- Selecting an external editor.

Figure 9.3.: The Options drop down item

9.4.1. Table Editor

The following sections deal with the management of feature codes to be used in code conversion tables. Since all this information is stored in database tables, a general description of the *Table Editor*, an instrument to edit database tables, follows.

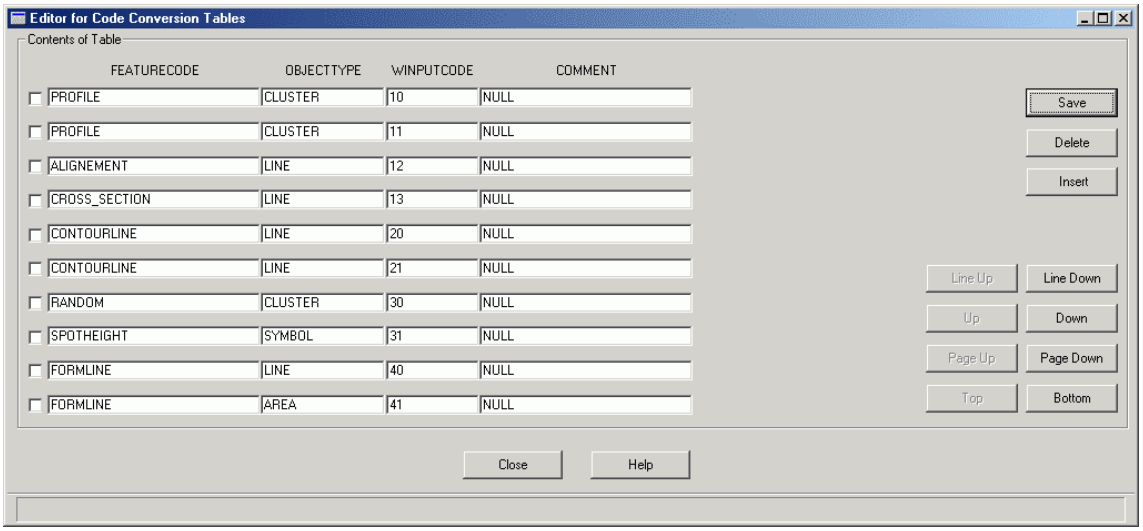


Figure 9.4.: Editor for Conversion tables window

In the central area of the table editor window the contents of the database table are displayed. If not all data sets can be displayed in a single screen the navigation buttons on the right margin of the window get active.

Click button

- *Line Down* to scroll one data set down (towards the end of the list)
- *Line Up* to scroll one data set up (towards the beginning of the list)
- *Down* to scroll five elements down
- *Up* to scroll five elements up
- *Page Down* to scroll a screen page down (9 data sets)
- *Page Up* to scroll a screen page up (9 elements)
- *Bottom* to scroll to the end of the list
- *Top* to scroll to the beginning of the list

Besides scrolling the table the dialog offers to

- insert,
- change or
- delete

data sets.

Inserting Data Sets

Clicking at state-switch *Insert...* a dialog window will open.

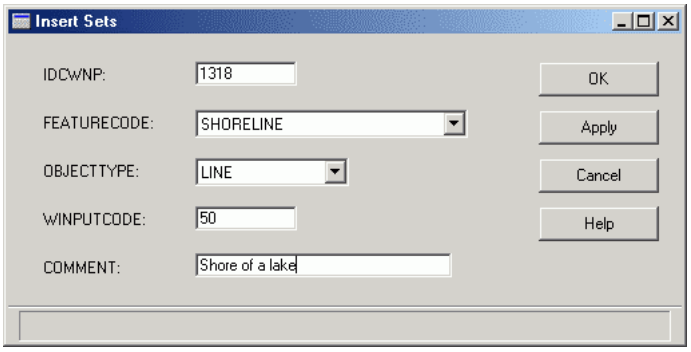


Figure 9.5.: Insertion of data sets window

The *Insert Window* contains either a text field or a selection for each column. In the latter case, the desired item can be selected from a list of valid items. If no such list is available a value can be entered directly into the text field . The columns *Remark* are optional and therefore need not be filled out.

To insert a single data set click the button *Ok*. The data set will be inserted into the database table and the window will be closed. To insert more than one data set click the button *Apply*. The window will remains open for to specify the next data set. To discard the specification of the data set press button *Cancel*.

Changing Data Sets

To update an existing element in a database table the desired element (row and column) has to be selected by a left mouse click or by tabbing to it. Once the element is selected a new value can be entered or selected from a list of available values. Updating can only be done interactively using the elements of the GUI in the current version, but not from within the command line (command file).

Deleting Data Sets

To delete an existing data set from a database table click the checkbox at the very left margin of the data set and press button *Delete*. Multiple data sets can be deleted in one step selecting the appropriate checkboxes . Removing data sets can only be done interactively in the current version, but not from within the command line (command file).

The changes of an editing session can be committed clicking at button *Save*. Pressing the button *Close* will end the editing session and close the window. Note that if changes have not been saved before pressing the button *Close* a dialog window will appear, asking whether or not to save the changes. Select *Yes* to save or *No* to discard the changes.

9.4.2. Feature Codes

As described in section 5.3.10 the coding information of data object is stored in the two columns *Featurecode* and *Objecttype* in the database.

While the list of valid object types is fixed and cannot be changed it is possible to add new feature codes to the list of predefined codes.

To manipulate feature codes click drop-down menu *Options* and select drop down dialog *Featurecodes...*. Thus the database table editor window opens (since feature codes are also stored in a database table).

The central area of this window initially contains all predefined feature codes (column KEYWORD) and for each feature code an optional comment (column REMARK). Deleting, changing or inserting of feature codes is done as described in section 9.4.1.

The insertion of new feature codes can either be done interactively using the elements of the GUI or from within the command line (command file).

cmL example

```
Options/
  Featurecodes/
    Insert,
      Keyword=(SHORELINE),
      Remark=(Shore line of a lake)
      Ok;
    Save,
    Close;
```

All the specified feature codes (pre- or user defined) can be used to create user defined conversion tables. Note that the predefined feature codes should not be manipulated!

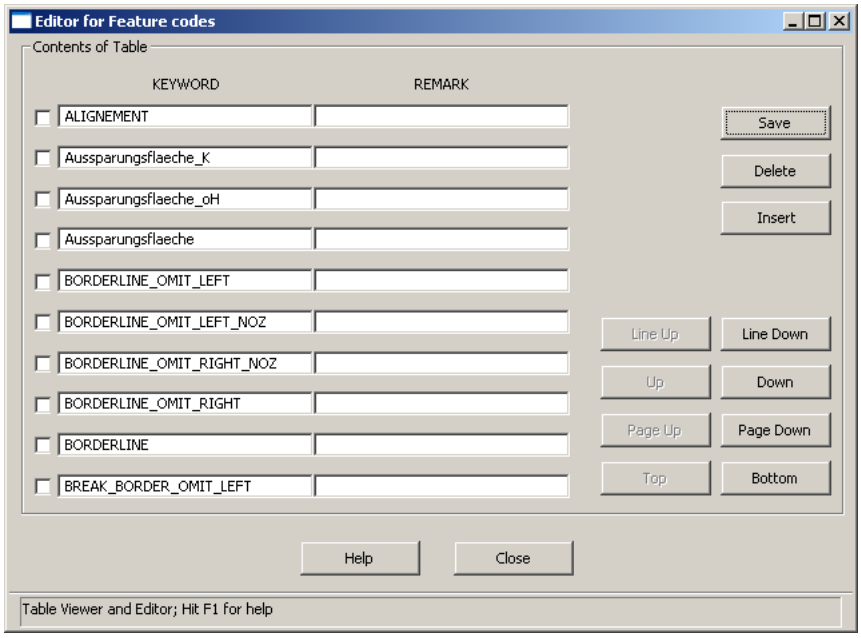


Figure 9.6.: Editor window for feature codes

9.4.3. Conversion Tables

All operations concerning the management of code conversion tables can be accessed by selecting the drop down dialog *Conversion tables...* from within the drop-down menu *Options*.

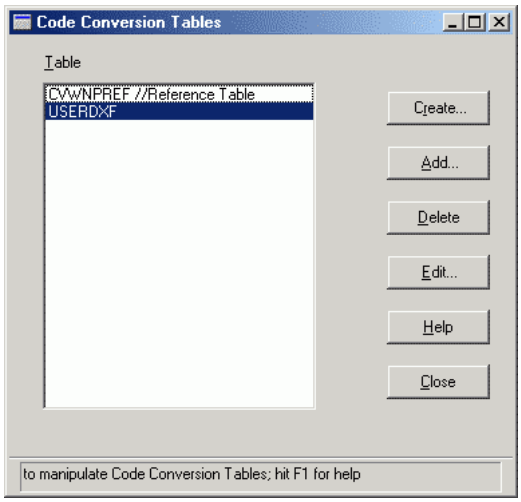


Figure 9.7.: Conversion tables window

The window contains the list of all user defined conversion tables as well as the reference table (please refer to section 5.3.10 for general information about data coding and conversion tables). Furthermore a series of buttons for different operations is available.

These buttons are to

- create a new (empty) user defined conversion table
- add an existing user defined conversion table to the database catalog
- edit the contents of a conversion table (user defined or reference table)
- delete an user defined conversion table
- close the window

Creating a New Conversion Table

To create a new user defined conversion table press state-switch *Create...* from within the window *Code conversion tables*. The following window will open.

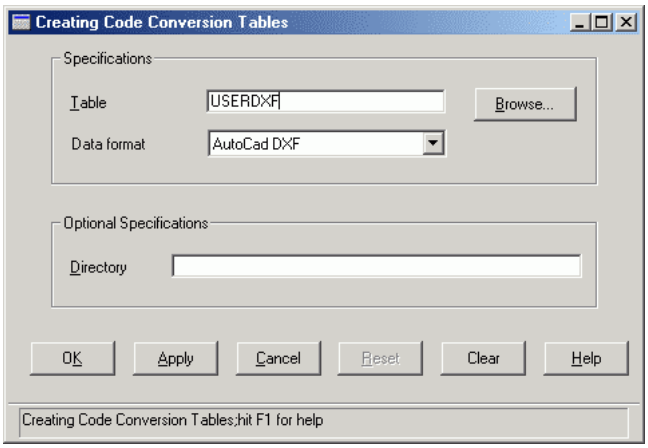


Figure 9.8.: Creating code conversion tables window

The name of the new code conversion table has to be entered in the text field *Table*. The name may be up to eight characters long and may contain the letters A..Z and the digits 0..9. Lower case letters will automatically be converted to upper case. The data format for which a conversion table shall be created can be select from the list available data formats. The specification of a directory is optional. If no directory is specified (recommended), then the conversion table is created in the directory

```
<ScopProjects>\topdb\cvt
```

, where

```
<ScopProjects>
```

is the path of the SCOP Projects directory specified at installation time.

To create the conversion table press either button *Ok* or button *Apply*. The appropriate table will be created and inserted in the database table catalog. Furthermore the name of the table will appear in the list of available tables in the window *Code conversion tables*. Note that the table still is empty and needs be filled as described in 9.4.3. To discard the operation press button *Cancel*.

cmL example

```
Options/
  ConvTables/
    Create,
      Table=(USERDXF),
      Dataformat=("AutoCAD DXF"),
      Ok;
```

Adding an Existing Conversion Table

Adding is eg. useful to overtake a conversion table from a different SCOP++ installation. Precondition for adding a conversion table is, that the respective table exists on the disk.

To add an existing user defined conversion table press state-switch *Add...* from within the window *Code conversion tables*. The following window will open.

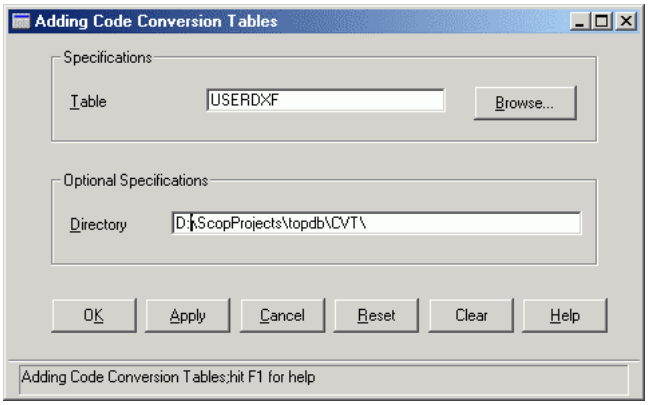


Figure 9.9.: Adding code conversion tables window

Within the window *Adding code conversion tables* the name of the (existing) conversion table has to be specified. This can either be done by inserting the name (without the file extension.TOP) in the text field *Table* and the complete path in the text field *Directory* or by browsing for the respective conversion table file. If no directory is specified, then the conversion table is searched in the directory

```
<ScopProjects>\topdb\cvt
```

, where

```
<ScopProjects>
```

is the path of the SCOP Projects directory specified at installation time. It is recommended to place all conversion tables in this directory.

Pressing button *Ok* or button *Apply* the specified table will be added to the database table catalog. Furthermore the name of the table will appear in the list of available tables in the window *Code conversion tables*.

After adding a conversion table a consistency check is performed. Please refer to section 9.4.3 for more details.

Editing a Conversion Table

To edit a conversion table choose the desired table from the selection by clicking at it and press state-switch *Edit...*.

Figure 9.6 shows the table editor dialog for the reference conversion table (CVWNPREF). In the central area of the window the contents of the table are displayed, containing the columns *Featurecode*, *Objecttype*, *Winputcode* and *Remark*. The column *Winputcode* is only available for conversion tables of WINPUT data format. Conversion tables of other data formats would have other columns instead (eg. DXF: Layer, Entity). Please refer to section 5.3.10 for general information about data coding.

Updating, deleting and editing data sets is described in section 9.4.1. In the following examples for the insertion of a singe data set for each data format using the command language (cmL/cmF) are given.

cmL example

```
Options,
  ConvTables,
    Table=(CVWNPREF),
    Edit.../
      Insert,
        Featurecode=(Shoreline),
        Objecttype=(LINE),
        Winputcode=(50),
        Remark=(shore of a lake),
        Ok;
```

cmL example

```
Options,
  ConvTables,
    Table=(USERDXF),
    Edit.../
      Insert,
        Featurecode=(Shoreline),
        Objecttype=(LINE),
        Layer=(Shore),
        Entity=(POLYLINE),
        Remark=(shore of a lake),
        Ok;
```

cmL example

```
Options,
  ConvTables,
    Table=(USERAIG),
    Edit.../
      Insert,
        Featurecode=(Shoreline),
        Objecttype=(LINE),
        SFT=(3),
        Remark=(shore of a lake),
        Ok;
```

cmL example

```
Options,
  ConvTables,
    Table=(USER),
    Edit.../
      Insert,
        Featurecode=(Shoreline),
        Objecttype=(LINE),
        Remark=(shore of a lake),
        Ok;
```

Note that XYZ data format doesn't contain any coding information. But nevertheless a conversion table containing all relevant Featurecode/Objecttype coding pairs must exist. In this way the amount of data can be controlled, when exporting to XYZ format.

Please keep in mind that the FEATURECODE/OBJECTTYPE coding pairs don't need to be unique within a code conversion table and the uniqueness is therefore not checked by the program. This implies that the user himself is responsible for the correctness of a conversion table.

After editing a conversion table a consistency check is performed. Please refer to section 9.4.3 for more details.

Deleting a Conversion Table

To delete a conversion table choose the desired table from the selection by clicking at it and press button *Delete*. The table will be removed from the database catalog and the appropriate file will be removed from the computers hard disk.

To close the window *Conversion tables* press button *Close*.

Consistency Check for Conversion Tables

As described in section 5.3.10 the reference conversion table (CVWNPREF) is used for the preparation of input data in WINPUT data format for the several SCOP server processes. Thus, this table must contain a complete set of featurecode/objecttype coding pairs and

Table 9.1.: Default WINPUT codes

Objecttype	WINPUT Code
CLUSTER	30
SYMBOL	31
LINE	50
AREA	51
POINT	30

the appropriate WINPUT code for each combination. After adding or editing a conversion table this precondition may not be fulfilled. For this reason a consistency check is performed by SCOP++ each time a conversion table is added or edited.

This check scans all relevant conversion tables, searches all featurecode/objecttype coding pairs and checks whether each combination is available in the reference conversion table. If a coding pair is missing, this data set is automatically inserted into the reference conversion table using default WINPUT codes and an information window is displayed on the screen. Table 9.1 shows the default assignment of WINPUT codes depending on the appropriate object type.

9.4.4. Protocol Options

The drop down dialog *Conversion tables...* is followed by the checkboxes for the protocol options (see figure 9.4):

Protocol : If the checkbox *Protocol* is checked the log files of some the calculation will be written to the disk. This file can be found in the directory of the current project with the name *projectName.log*. Protocolling is enabled per default, but mind that only protocols of DTM, slope, isolines ,calculation of difference models, volumes and algebraic models will be documented.

Error : If the checkbox *Error* id checked the error messages will be written to the disk. This file can be found also in the project directory of the current project with the name *projectName.err*. Protocolling of the error messages is enabled per default also.

9.4.5. Operation Mode Options

SCOP++ can be run in two different operation modes.

Interactive : If the checkbox *Interactive* is checked SCOP++ runs in interactive operation mode. This means that all error and information messages as well as ask user dialogs are displayed on the screen, requiring user interaction. Interactive is the default in any case except when starting SCOP++ in BATCH mode (command line option -MBATCH).

Unattended : If the checkbox (Unattended) is checked SCOP++ runs in unattended operation mode. This means that no error or information messages are displayed on

the screen. Furthermore the ask user dialogs will not be presented either. In this case the default answer (the left-most button of the dialog) is selected automatically. This mode is especially useful when command procedures (triggering long lasting calculations) are started from the command line (cmL) and the process is not attended by the user sitting in front of the computers terminal. In unattended operation mode SCOP++ runs exactly the same way as in batch mode (command line option -MBATCH).

9.4.6. Editor Preferences

This dialog specifies the the editor, which will be used for showing the log- and the error files. Per default the Notepad.exe is used, but you can specify any editor, which accepts a file to open as program argument.

9.5. Tools

The bar item *Tools* encloses general tools and all tools, which may be deployed for one overlay. Up to now these are:

System command : Submit operation system commands (see section 9.5.1).

Z tool : Quick point-wise inspection of the interpolated surface of an overlay (see section 9.5.2.

Interpolation/Check tool : compute the Z value of data points or inspection points from the interpolated surface of an overlay (see section 9.5.3.

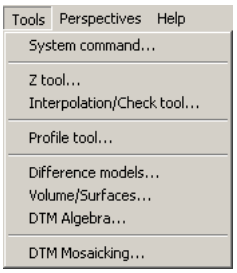
Profile tool : Quick profile-wise inspection of the interpolated surface of an overlay (see section 12.10.2).

Difference models : Calculation of difference models to other DTM's or horizontal planes (only available with the *SCOP++* Functionality Package "SCOP++ Analyzer"). See section 9.5.4).

Volume and Surface computations : Calculation of volumes or surfaces of parts or whole DTM's (only available with the *SCOP++* Functionality Package "SCOP++ Analyzer"). See section 9.5.5).

Arbitrary function models : Combination of two DTMs by an arbitrary function (only available with the *SCOP++* Functionality Package "SCOP++ Analyzer"). See section 9.5.6).

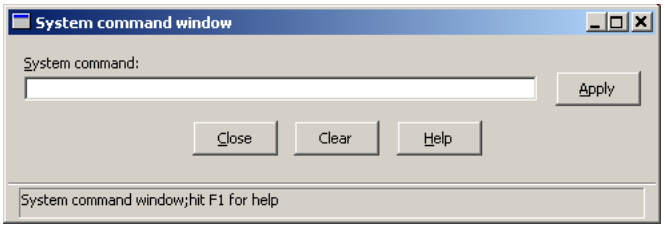
DTM Mosaicing : Mosaicing of DTMs at arbitrary boundary lines using some blending function to smooth out height differences (only available with the *SCOP++* Functionality Package "SCOP++ Analyzer"). See section 9.5.7).



Common for all entries (except System command) is that they can only be accessed if there already exists at least one calculated DTM for an overlay of the project. This can either be a model-only overlay or an overlay of type both, for which the DTM has been derived. In all these cases the items below the bar item *Tools* will be made accessible. Each of these tools has an selection *Select an overlay* in which all overlays of the project are listed and can be used to determine the overlay for which the tool shall be applied. The initial selection will be the element *NONE*, stating that no overlay has been selected. Selecting an overlay which is included in the project, but hasn't got any valid DTM (up to now) will yield a warning message and the selection will be set to the previous state.

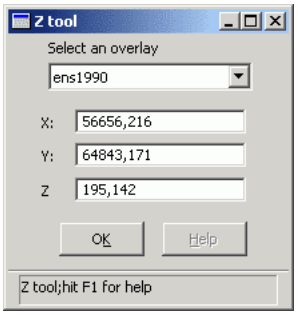
As the tools require the access of the DTM file and this resource has to be shared with other processes (eg. the derivation of views, the re-interpolation of the model and the similar), this can lead to a conflict. To obviate this, all the tools have lowest priority concerning DTM access which will be given up in favour of any other process. So if any other process has to access the DTM file this windows will be closed.

9.5.1. System Command



This simple dialog is designed to submit operating system commands. This may be useful especially in batch mode, when it may be become necessary to copy or move some result files to a certain location or the similar. To submit an operating system command enter the command in the text field *System command:* and press either <RETURN> or button *Apply*. To close the dialog press button *Close*.

9.5.2. Z Tool



After selecting an overlay with an available surface the Z values will be interpolated and displayed in numeric field Z when moving the cursor over a valid DTM area of the overlay. Of course the X and Y values will be updated also. If the cursor-coordinates haven't any valid Z value (there is no valid DTM at this location), the Z value will be set to zero.

9.5.3. Interpolation/Check Tool

The *Interpolation/Check tool* allows to interpolate elevations (Z values) by means of a DTM for terrain points given only by their plan coordinates. If elevations of these data points are available too then the differences to the elevations interpolated from the DTM are checked. In order to accomplish this task a DTM must be available. The DTM to be used for this purpose has to be chosen using the selection *Select an overlay* (see fig. 9.10). The next decision the user has to make is to select a *Source* for the data points. Five options are available:

- loading point coordinates from a *File*,
- using grid points from a reference *Overlay*,
- entering data manually using the *Keyboard*,
- digitizing points in the main graphics panel using the mouse *Cursor* or,
- digitizing points of the object in the main graphics panel using the mouse *Select*.

Depending on the above mentioned selection further relevant input fields become active and ready to enter necessary data, while other fields not used in conjunction with this selection will be deactivated (displayed in grey). A user may specify an input file and data format, an output file and data format, and on demand some statistics will be written to the logfile of the project.

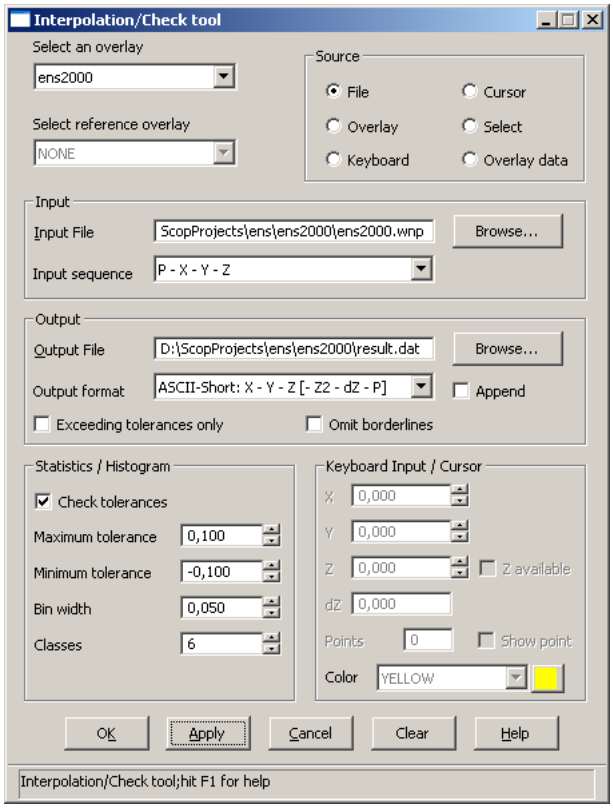


Figure 9.10.: Interpolation/Check Tool

Source option *File*

If the source option *File* is chosen, coordinates of points are loaded by the program from a file. This option requires to specify a filename within group *Input*. The user may type a filename and optional a path in the text field *Input file* or use the button *Browse...* to get a file browser for selecting a file. Additionally the proper data format for the file must be specified using selection *Input sequence*. Possible options are:

X - Y
Y - X
P - X - Y
P - Y - X
X - Y - P
Y - X - P
X - Y - Z
Y - X - Z
X - Y - Z - P
Y - X - Z - P
P - X - Y - Z
P - Y - X - Z

where

X . . . east coordinate,
Y . . . north coordinate,
Z . . . elevation,
P . . . point number.

The result of the calculation is saved by the program in the output file, whose name and path should be specified by the user within group *Output* in the text field *Output File*. It

is also possible to select a file using a file browser by activating the button *Browse...*. The user should remember that when choosing an existing output file its content is lost and overwritten by new data unless the checkbox *Append* is selected. Activating checkbox *Append* allows to add new data at end of the file without loosing old data. Additionally the proper format for the output file can be specified using the selection *Output format*. The following formats are available:

ASCII-short: X - Y - Z [- Z2 - dZ - P] ... max. 10 digits per datum
ASCII-long : X - Y - Z [- Z2 - dZ - P] ... max. 13 digits per datum
SCOP Winput

The user can choose one of the above mentioned formats with selection *Output Format*. The values in the square brackets are optional and calculated only if a Z value is available from the input data. When selection *Output Format* SCOP Winput is selected and Z values are available from the source then Z differences will be written to the text field *Output File* instead of interpolated Z values.

Selecting the checkbox *Omit borderlines* allows the interpolation of all points of the DTM regardless of borderlines present in the DTM. A further control of the the output is possible with checkbox *Exceeding tolerances only*. If this is selected, the result is checked against the given *minimum tolerance* and *maximum tolerance* from group *Statistics / Histogram* and written to the output file only if the value dZ is outside of the specified tolerance range. The button *Exceeding tolerances only* is available if the checkbox *Check tolerance* within group *Statistics / Histogram* is selected.

If the checkbox *Check tolerances* is selected, some statistics values are calculated and written to logfile of the project. This includes standard error, mean error, maximum of dZ and minimum of dZ and a histogram which may be further specified by numeric fields *bin width* and *classes*. If the number of classes n is specified, the range between *minimum tolerance* and *maximum tolerance* is cut into $n - 2$ classes and two classes complement the range to $-\infty$ and $+\infty$. The *bin width* is calculated automatically. If on the other side a *bin width* is specified by the user, then the number of classes will be calculated. When starting the process the given *bin width* is adapted in such a way that a whole number of classes can be placed within the given tolerance range.

If *check tolerances* is selected the following information is written to the logfile of the project:

```
Control point interpolation/check
Selected overlay  : ens2000
Input file        : D:\ScopProjects\ens\ens2000\check_in.xyz
Output file       : D:\ScopProjects\ens\ens2000\check_out.xyz

Number of points    :    3805
Standard error (rms):       1.932
Mean error          :      -0.108
Maximum             :      85.490
Minimum             :      -5.600

Interpolation failed for:       1 point(s)
Tolerances exceeded for :    2890 point(s)

Distribution of Z differences:
```

```
------------------------------
          dZ <  -0.100 :      1547    40.7 %
-0.100 <= dZ <  -0.060 :       127     3.3 %
-0.060 <= dZ <  -0.020 :       193     5.1 %
-0.020 <= dZ <   0.020 :       298     7.8 %
 0.020 <= dZ <   0.060 :       164     4.3 %
 0.060 <= dZ <   0.100 :       133     3.5 %
 0.100 <= dZ           :      1343    35.3 %
```

Before starting the calculation the program automatically checks the completeness of the specifications. To launch the calculation the following fields must by filled in or selected (otherwise the program suspends the calculation and reports an error):

Select an Overlay
Input File
Input Sequence
Output File

The calculation is triggered after clicking one of the buttons *Apply* or *OK*. Button *OK* additionally closes the dialog window.

A commandline / commandfile example for this window and the source option *File* is:

cmL example

```
// Points interpolation from the file
@InterpFile;
 Tools,
    Interpol/
      // Assuming there is an overlay ens2000
      SelectanOverl=(ens2000),
      InterpolationCheck=File,
      InputF=(D:\ScopProjects\ens\ens2000\inInter.wnp),
      OutputFi=(D:\ScopProjects\ens\ens2000\outInter.wnp),
      OutputFo=(SCOP),
      Omit=0,
      Checktol=1,
      Exceed=1,
      Max=0.200,
      Class=6,
      OK;
@endProc;
```

Source option *Overlay*

If the source option *Overlay* is selected then the inspection points are taken from an overlay selected with the selection *Select reference overlay*. The grid points of the reference overlay will be used as check points. The following fields have to be filled in or properly selected so that the program can start performing calculations:

Select an Overlay
Select Reference Overlay
Output File

Otherwise the programme will respond with an error.

For the description of group *Output* as well as group *Statistics / Histogram* see section 9.5.3.

A commandline / commandfile example for the source option *Overlay* is:

cmL example

```
// Points interpolation from the reference overlay
@InterpOver;
 Tools,
    Interpol/
      // Assuming there is an overlay ens2000
      SelectanOverl=(ens2000),
      InterpolationCheck=Overlay,
      Selectref=(ens2001),
      OutputFi=(D:\ScopProjects\ens\ens2000\outInter.wnp),
      OutputFo=(ASCII-L),
      Omit=1,
      Checktol=0,
      OK;
@endProc;
```

Source option *Keyboard*

If the source option *Keyboard* is selected then the inspection points can be entered in the numerical fields of the group *Keyboard Input / Cursor* by using a keyboard. When pressing button *Apply* the value of Z will be interpolated and written to numeric field Z. If the checkbox Z *available* is selected a value may be entered into the numeric field Z too. In this case the interpolated value of Z is not written to numeric field Z, but the difference $Z_{data} - Z_{DTM}$ will be displayed in the numeric field dZ. The numeric field *Points* shows the number of points interpolated since opening the window. This value will be reset to zero if the window is closed.

The interpolated points obtained by keyboard input may be written to a file if the user specifies an *Output File*. Option *Append* is automatically selected in this case. The *Output File* will be not be written until clicking on the button *OK*.

This option requires the selection of an overlay and at least the numeric fields X and Y have to be filled in.

For the description of group *Output* as well as group *Statistics / Histogram* see section 9.5.3.

A commandline / commandfile example for the source option *Keyboard* is:

cmL example

```
// Points interpolation by the use of keyboard
@InterpKeyb;
 Tools,
    Interpol/
      // Assuming there is an overlay ens2000
      SelectanOverl=(ens2000),
      InterpolationCheck=Keyboard,
      OutputFi=(D:\ScopProjects\ens\ens2000\outInter.wnp),
      OutputFo=(ASCII-L),
      X=57002.628,
      Y=64670.407,
      Zavailable=1,
      Z=255.300,
      Apply,
      X=57176.386,
      Y=64684.525,
      Z=259.900,
      Apply,
      X=57315.392,
      Y=64722.534,
      Z=217.700,
      Apply,
      OK;
@endProc;
```

Source option *Cursor*

The source option *Cursor* enables the input of point data by mouse click. The main graphics panel is will be set to a mode which allows digitizing of points and hand over them to the *Interpolation / Check Tool*. In this mode clicking the left mouse button while the mouse pointer is located within the area of the main graphics panel will transfer the coordinates of the digitized point to the numeric fields *X* and *Y* then the elevation is interpolated and displayed in numeric field *Z* within group *Keyboard Input / Cursor*. Each click of a left mouse button introduces coordinates of the new interpolated point, provided the main graphics panel remains in the the mode "digitize point". The mode "digitize point" will be switched off after pressing one of the following buttons: *OK, Apply, Cancel, Clear* or after activating a radio button other than *Cursor* within the group *Source*. The selected mode *Cursor* may also be cancelled by clicking the right mouse button over the main graphics panel. The numeric field *Points* shows the number of points interpolated since opening the window. This value will be reset to zero if the window is closed.

The interpolated points obtained by cursor input may be written to a file if the user specifies an *Output File* within group *Output*. Option *Append* is automatically selected in this case. The *Output File* will be not be written until clicking on the button *OK*.

For the description of group *Output* as well as group *Statistics / Histogram* see section 9.5.3.

A pre-condition for his option is the selection of an overlay with selection *Select an overlay*.

A commandline / commandfile example for the source option *Cursor* is:

cmL example

```
// Points interpolation by the use of cursor
@InterpKeyb;
 Tools,
    Interpol/
      // Assuming there is an overlay ens2000
      SelectanOverl=(ens2000),
      InterpolationCheck=Cursor,
      OutputFi=(D:\ScopProjects\ens\ens2000\outInter.wnp),
      OutputFo=(ASCII-L),
      showp=1,
      omit=0;
   ens2000,
     Data,
       displ;
@endProc;
```

Now press you please button *OK* in the *Interpolation/Check tool* dialog to terminate the program.

Source option *Select*

The source option *Select* is similar to the option *Cursor* and enables the input of many points data which belong to the object by mouse click. The main graphics panel is will be set to a mode which allows digitizing of points of the data object and hand over them to the *Interpolation / Check Tool*. The description of this window behavior applies to the source option *Cursor*.

A commandline / commandfile example for source option *Select* is:

cmL example

```
// Points interpolation by the selecting objects
@InterpSel;
 Tools,
    Interpol/
      // Assuming there is an overlay ens2000
      SelectanOverl=(ens2000),
      InterpolationCheck=Select,
      OutputFi=(D:\ScopProjects\ens\ens2000\outInter.wnp),
      OutputFo=(ASCII-L),
      omit=1,
      Checktol=1,
      Exceed=1,
      Max=0.01,
      Class=4;
   ens2000,
     Data,
       displ;
@endProc;
```

Now press you please button *OK* in the *Interpolation/Check tool* dialog to terminate the program.

Source option *Overlay data*

The source option *Overlay data* is similar to the option *Overlay*. But the original data of the reference overlay will be used as check points.

The following fields are mandatory:

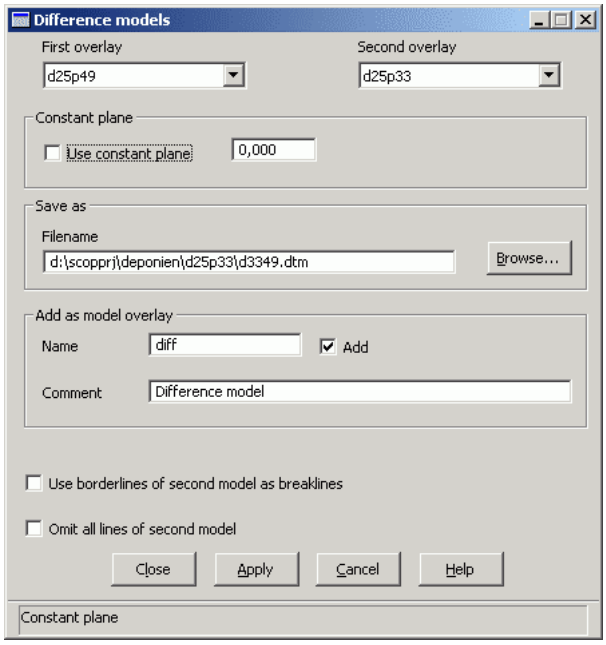
Select an Overlay
Select Reference Overlay
Output File

Otherwise the programme will respond with an error.

For the description of group *Output* as well as group *Statistics / Histogram* see section 9.5.3.

9.5.4. Difference Models

This window serves for the purpose of combining two DTMs. Although this combination generally can be performed by proliferation of an arbitrary function, in this case it is restricted to the subtraction of two DTMs. The second model doesn't even need to be a real DTM, it can also be specified as horizontal plane of given height. The result of the combination is a third, new digital model. It is called function model. The function model has the structure, i.e. the grid size and the borderline information, of the first input



model. Breakline and formline information of both input models are integrated into the function model.

The selection *First overlay* and the selection *Second overlay* help to choose from the overlays of the current project. Their contents comprise all overlays for which a DTM is available. Generally, it is not important which one of the two input models is the first model and which one is the second model. But two facts might result in a loss of information, if the order of the input models is not chosen carefully:

1. When the first DTM has a larger grid size than the second one, the resulting DTM gets the lower of the two possible resolutions, because the structure of the first model is taken for structure of the function model.
2. Furthermore, the borderlines of the second model are disregarded, in order to avoid contradictions of borderlines. This behaviour can be changed by selecting the checkbox *Use borderlines of second model as breaklines*

In the group *Save as* the filename of the resulting DTM can be specified. Note that the extension .dtm will be appended if omitted. Also the path will be amended to the overlay directory of the second overlay (or the first overlay if the usage of a horizontal plane is chosen - see below), if no path is specified.

In the case the checkbox *Add* in the group *Add as model overlay* is activated and the text field *Name* is not empty, a new model overlay of type model-only will be added to the current project.

If a horizontal plane instead of the second model should be used the checkbox *Use constant plane* has to be selected. This action makes the selection *Second overlay* inaccessible and enables the input of the Z value of the plane in the numeric field right to it.

The two checkboxes at the bottom of the window deal with the propagation of line information to the function model.

- Selecting the checkbox *Use borderlines of second model as breaklines* introduces the border lines out of the second model are as breaklines into the resulting function model, whereas
- Selecting the checkbox *Omit all lines of second model* discards all of them.

Pressing the button *Apply* triggers the calculation. Note that it will be remain inaccessible after a calculation as long as there aren't any changes of the parameters.

A commandline / commandfile example for this window is:

cmL example

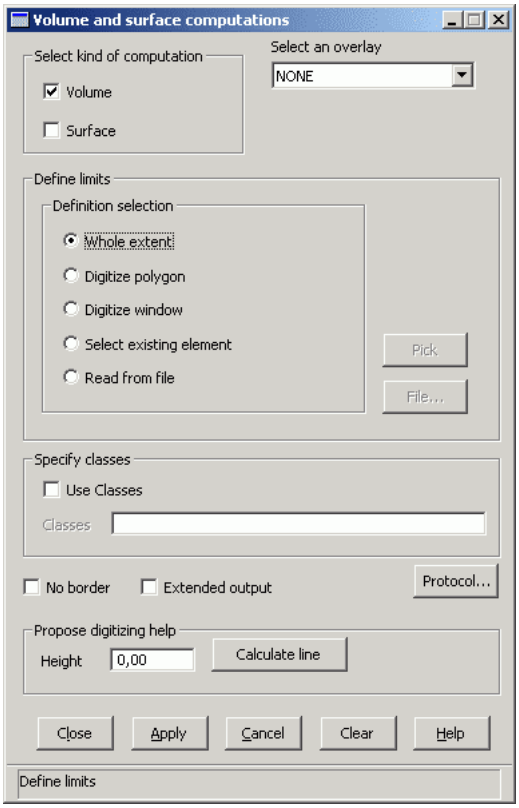
```
@Diff;
// Calculate difference models and insert them
// as overlays of type model-only:
// (1) p33 - p49
// (2) p33 - horizontal plane with height 315
Tools,
  DifferenceModel/,
    UseConst=(n),
    First=(d25p33),
    // It's assumed there is an overlay
    // named d25p33 available
    Second=(d25p49),
    // It's assumed there is an overlay
    // named d25p49 available
    FileName=(d33_49),
    Add=(y),
    Name=(d33_49),
    Comment=(Difference model of two model overlays),
    UseBorderlines=(y),
    Apply,

    UseConst=(y)(315),
    FileName=(d315),
    Add=(y),
    Name=(d315),
    Comment=(Difference model with constant plane),
    UseBorderlines=(y),
    OK;
@endProc;
```

9.5.5. Volume and Surface Computations

This window serves for calculation of volumes and surfaces of DTMs for specified areas.

Please determine the kind of computation via the checkbox *Volume* and the checkbox *Surface* and the DTM via the selection *Select an overlay*. This selection comprises all overlays for a DTM is available.



The areas can be specified via means of the group *Define limits*. There you have several possibilities to define the limits by selection one of the radio buttons on the right:

- Whole extent takes the whole area of the overlay ie. the whole area where a valid DTM for the selected overlay exists.
- Digitize polygon serves for digitizing a number of polygons in the main graphics panel. After pressing the button *Pick* one polygon can be digitized. Several polygons can be digitized by successively pressing the button *Pick* and digitizing. Open polygons will be closed. The button *Pick* will be made unaccessible during digitizing and made accessible again afterwards. If the digitizing is interrupted by any other digitizing or modification action of elements in the main graphics panel the button *Pick* has to be pressed again.
- Digitize window servers for digitizing axe parallel windows in the main graphics panel. Here the same mechanism for digitizing more than one window applies as above.
- Select existing element makes it possible to select an element in the main graphics panel.
- Read from file serves for reading the limits from file. The file can be browsed for with the file-browser beneath the button *Browse...*. The following file formats will be accepted:

- a SCOP Limits file (ASCII) conforming to the following format:

```
Line Contents
-----------------------------------------------------------
1    Number of polygons
2    Area number of the first polygon area, and sub
     area number of the first polygon area
3    Number of points belonging to the first limitline
4    Easting and northing of first point 5-X
     "       "   "        "
    further points X+1

For further polygons see line 2ff.
-----------------------------------------------------------
```

  The last point of a limitline is connected to the first point by the program. So it is not necessary to store the first point of a line at its end again.

- a simple XY or XYZ file (only one polygon),
- a WINPUT file,
- a DXF file.

The raster model is subdivided into classes of different z-values. For each class the results (surfaces,volumes) are calculated. When computing volumes the user normally is interested in fill and cut. This problem is solved by creating a height difference model and disabling the usage of classes by deselecting the checkbox *Use classes*, thus subdividing the z-coordinates in two groups with negative or positive values. This is also the default value.

If the usage of classes is specified, it must be at least one class value.

Selecting the checkbox *No border* ignores the borderlines are ignored. This means that excluded areas are used in the computation, if they are included in the DTM.

Selecting the checkbox *Extended output* extends the output information. A summary of this output - the calculation results - can be inspected in the window beneath the state-switch *Protocol...*. The whole output will be visible in the log file of the project (see section 9.4.4).

Please note that the volume calculation of difference models isn't that easy. That comes because of the fact that the contour-line with height zero is only poorly defined in areas which have the same surface in both of the original DTMs. Therefore it is often necessary to manually define the areas of interest for the volume computation (ie. the areas where a removal or application is supposed). This can be facilitated by superimposing an orthophoto during digitalisation (see section 13) or by trying to calculate the contour-line with height zero and manually correcting it. In the group *Propose digitizing help* you can specify the height of the contour-line in the numeric field *Height*. After pressing the button *Calculate* the contour-line will be inserted in the main graphics panel. Successively calculations will removed the previously calculated contour-lines before their insertion.

A commandline / commandfile example for this window is:

cmL example

```
@Vol;
// Calculate volume and surface for overlay named d315.
// First for the whole area of the DTM and then for a
// limited area read from a limits file.
Tools,
  VolumeSurf/,
    Volume=(y),
    Surface=(y),
    SelectAnOverlay=(d315),
    WholeExtent,
    UseClasses=(y),
    Classes=( -10 0 10 20 30 ),
    Apply,

    Clear,
    ReadFromFile,
    File,
      FileName=(d315_alg.lim),
      // Default directory is project directory!
      ok;
    Apply;
@endProc;
```

9.5.6. DTM Algebra

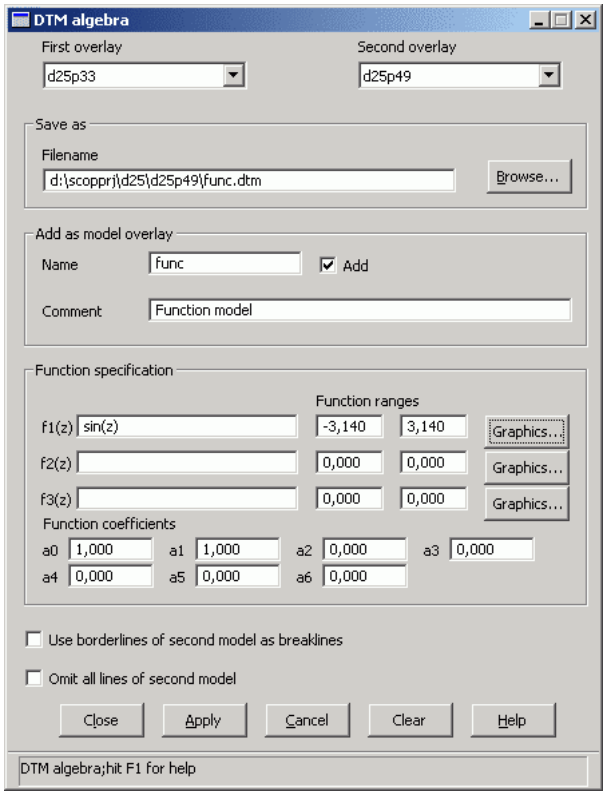
This window serves for combination of two DTMs by arbitrary functions. It can be seen as an extension to ordinary difference models (see section 9.5.4).

The result of the combination is a third, new digital model. It is called function model. The function model has the structure, i.e. the grid size and the borderline information, of the first input model. Breakline and formline information of both input models are integrated into the function model.

The function instruction is intended to be applicable to a wide range of different tasks. It has the general form:

$$\begin{aligned} z_3 = \quad f_3(\quad & a_0 + a_1 * f_1(z_1) + a_2 * f_2(z_2) + a_3 * f_1(z_1) * f_2(z_2) + \\ & a_4 * \tfrac{f_1(z_1)}{f_2(z_2)} + a_5 * \tfrac{f_2(z_2)}{f_1(z_1)} + \tfrac{a_6}{f_1(z_1) * f_2(z_2)}) \end{aligned} \tag{9.1}$$

z_1 and z_2 are the z-values of the first and the second input model. The functions f_i and the coefficients a_i are user definable. Generally, it is not important which one of the two input models is the first model and which one is the second model. But two facts might result in a loss of information, if the order of the input models is not chosen carefully:



1. When the first DTM has a larger grid size than the second one, the resulting DTM gets the lower of the two possible resolutions, because the structure of the first model is taken for structure of the function model.
2. Furthermore, the borderlines of the second model are disregarded, in order to avoid contradictions of borderlines. However, with parameter checkbox *Use borderlines of the second model as breaklines*, these lines may be used as breaklines in the function model.

In the group *Save as* the filename of the resulting DTM can be specified. Note that the extension .dtm will be appended if omitted. Also the path will be amended to the overlay directory of the second overlay (or the first overlay if the usage of a horizontal plane is chosen - see below), if no path is specified.

In the case the checkbox *Add* in the group *Add as model overlay* is activated and the text field *Name* is not empty, a new model overlay of type model-only will be added to the current project.

In general, two different digital models are combined according to formula above 9.1. First, the z-values of the input models are manipulated according to the functions f_1 and f_2. Then a term is computed according to equation 9.1 and the user defined coefficients. This term is argument to function f_3. The function value of f_3 is the z-value of the resulting model z_3.

The functions f_i and the coefficients a_i are defined in the group *Function specification*. You can enter the desired function in the corresponding text field and the ranges for the functions in the numeric fields left of it. Inspect the outcome in the graphics windows

beneath the button *Graphics...*. The graphics windows will be updated each time either the function definition or one of the range values change.

Following functions are recognized:

Mathematical	Keyword	Meaning
$+a$	+a	positive sign
$-a$	-a	negative sign
$a+b$	a+b	addition
$a-b$	a-b	subtraction
$a*b$	a*b	multiplication
a/b	a/b	division
$\sqrt{a}$	sqrt(a)	square root of a
a^2	a**2	a squared
$\sqrt[n]{a}$	a#n	n'th root of a
$\exp(a)$	exp(a)	exponential function of a
$\ln(a)$	ln(a)	natural logarithm of a
$\log(a)$	log(a)	logarithm to the base of 10 of a
$\sin(a)$	sin(a)	sine of a
$\arcsin(a)$	asin(a)	arcus sine of a
$\cos(a)$	cos(a)	cosine of a
$\arccos(a)$	acos(a)	arcus cosine of a
$\tan(a)$	tan(a)	tangent of a
$\arctan(a)$	atan(a)	arcus tangent of a
$\arctan(a)$	atan(a)	arcus tangent of a
$\partial(a)$	da(a)	partial derivation of a

After entering the function in the corresponding text field it will be interpreted and any interpretation errors will be marked. Empty function text field indicate that this function won't be used.

The calculation can be triggered by pressing the button *Apply*.

cmL example

```
@Func1;
// Function model 1
Tools/,
  DTMAlgebra/,
    FirstOverlay=(d25p33),
    // Assuming there is an overlay d25p33
    SecondOverlay=(d25p49),
    // Assuming there is an overlay d25p49
    Filename=(func),
    Add=(y),Name=(func),Comment=(Function model),
    f1=("ln(z)")(0.3)(10),
    f2=("1/z")(0.3)(10),
    f3=("dz(z**2)")(0.3)(10),
    // Don't take care if this makes sense
    a0 0.5 0.3 0.2 0.1 0.3,
    OK;
@endProc;
```

9.5.7. DTM Mosaicing

In certain cases the task may arise to merge two DTMs into a single combined DTM. Basically this is done best by retrieving the original data and re-interpolating the overall DTM. The interpolation algorithm of SCOP++ (linear prediction) is well suited for dealing with data from different data sources. Anyway in many cases the original data may not be at hand. For that reason the task is to combine both DTMs in such a way that the transit from the one DTM to the other is not noticeable. This process is often referred to as *'stitching', 'blending'* or - as in case of SCOP++ - *'mosaicing'*.

DTM mosaicing may be applied:

- to nest a small high accurate DTM into a larger DTM of lower accuracy (e.g. for visualization purposes)
- to homogenize the common edge of adjacent DTM tiles which have been calculated independently
- to replace existing DTM data with more recent data with respect to a smooth transit

In the following section the basic concept of deriving DTM mosaics is illustrated. A detailed description of all parameters of the respective dialog is given in the subsequent section.

Basic concept

The basic idea for achieving a smooth transit from one DTM to the other is to blend the z-values of both DTMs within a certain tolerance band in an appropriate way. Mathe-

matically this can be expressed as a weighted average of both z-values where the weights depend on the distance from the center of the tolerance band.

The tolerance band itself must surround the domain of the first DTM and may be given as a single closed boundary line. This line can either be the inner boundary, the outer boundary or even the center line of the band (the remaining elements of the band are calculated automatically). The mosaicing algorithm is capable of handling multiple boundaries (tolerance bands) in a single mosaicing process. Figure 9.11 shows a mosaic (DTM 1...eastern part, DTM 2...western part) which were blended along a single boundary line.

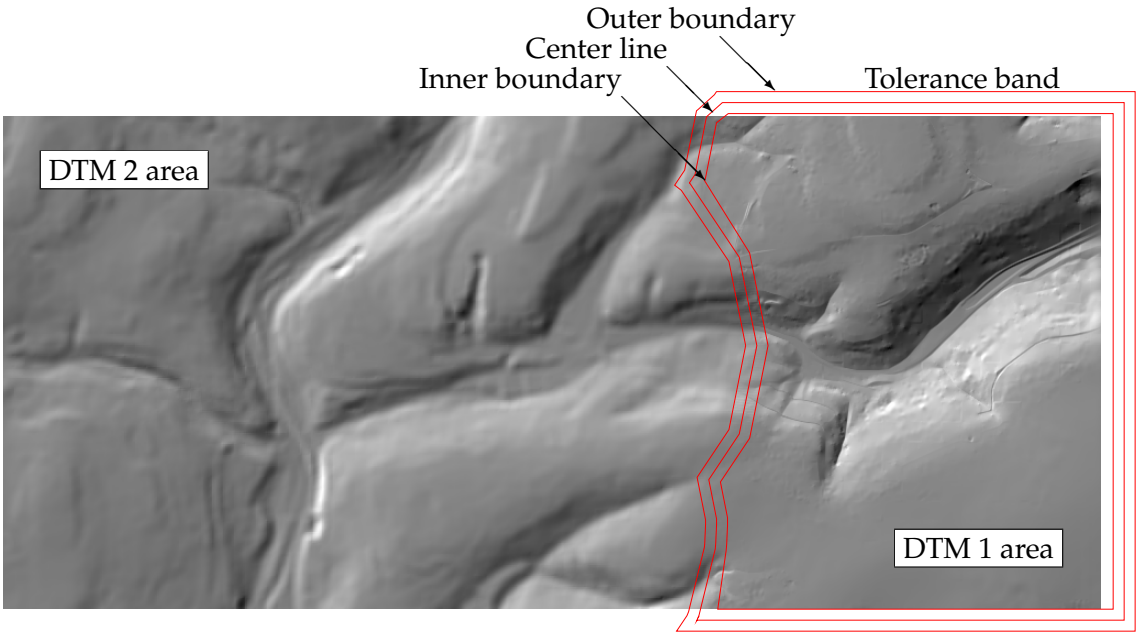


Figure 9.11.: Mosaicing two DTMs with different accuracy

Within the tolerance band the weight *w1* according to the 1st (inner) DTM starts at 0 and gradually increases to 1 at the opposite edge of the tolerance band. In the same way the weight *w2* according to the 2nd (outer) DTM decreases from 1 to 0. The z-values can then be calculated by the formula:

$$z = w_1 * z_1 + w_2 * z_2 \tag{9.2}$$

$$w_2 = 1.0 - w_1 \tag{9.3}$$

Different results can be achieved by applying different weight functions. Basically three types of weight functions can be distinguished.

- jump function
- linear function
- curved function

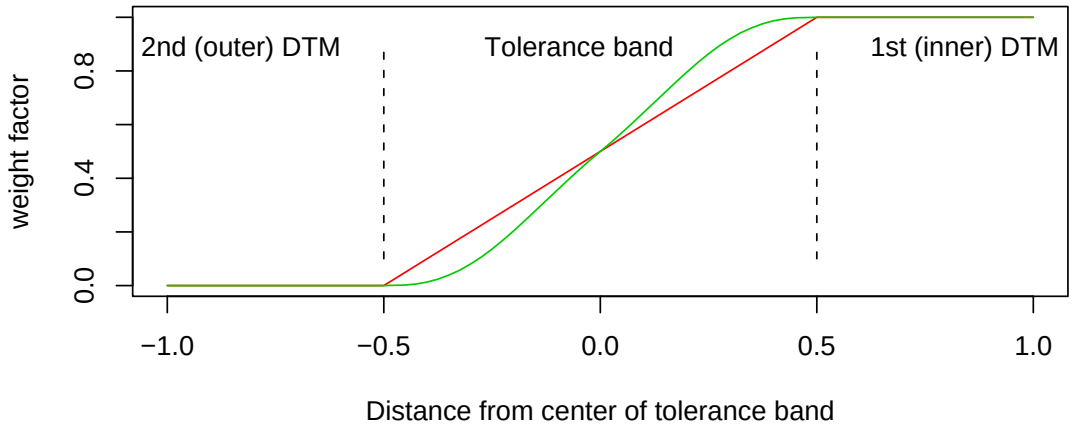


Figure 9.12.: Linear and curved weight function

Figure 9.12 shows that - regardless of the type - the weight function starts with 0 at the outer boundary of the tolerance band, increases to 0.5 in the center of the tolerance band and finally reaches 1 at the inner band edge. The advantage of the curved weight function is the horizontal tangent at the edges of the tolerance band, which causes a steady change of the elevations as well as the slopes. But this can only be achieved by a steeper increase of the weights (more abrupt transit) in the middle of the tolerance band. Using the linear weight function on the other hand ensures a continuous increase of the weights, which has proved to be best for most applications. Nevertheless the curved weight function may also be appropriate in some special cases. The jump function is a special case where the weights jump from 0 to 1 exactly in the middle of the band, which in fact means that there is no blending at all but a direct jump of the z-values from the 1st to the 2nd DTM.

Please note that the following preconditions have to be fulfilled for mosaicing of DTMs with SCOP++:

- The boundary line (tolerance band) must be given as a closed polygon
- Correct blending of the DTMs within the tolerance band can only be ensured if the z-values of both DTMs are available.
- The different tolerance bands may not overlap since this leads to unpredictable results in the mosaic.

Figure 9.13 shows three synthetic surfaces (tilted plane, roof and four-sided pyramid) and their merged combinations using the linear weight function.

DTM Mosaicing dialog

The following section contains a detailed description of all parameters necessary for deriving a DTM mosaic. For more general information about DTM mosaicing please refer to the previous section.

Initially the *First overlay* and *Second overlay* have to be specified from the list of available overlays. Definition of the *Operation mode* is the subsequent step. The *DTM mosaicing* dialog offers three different modes:

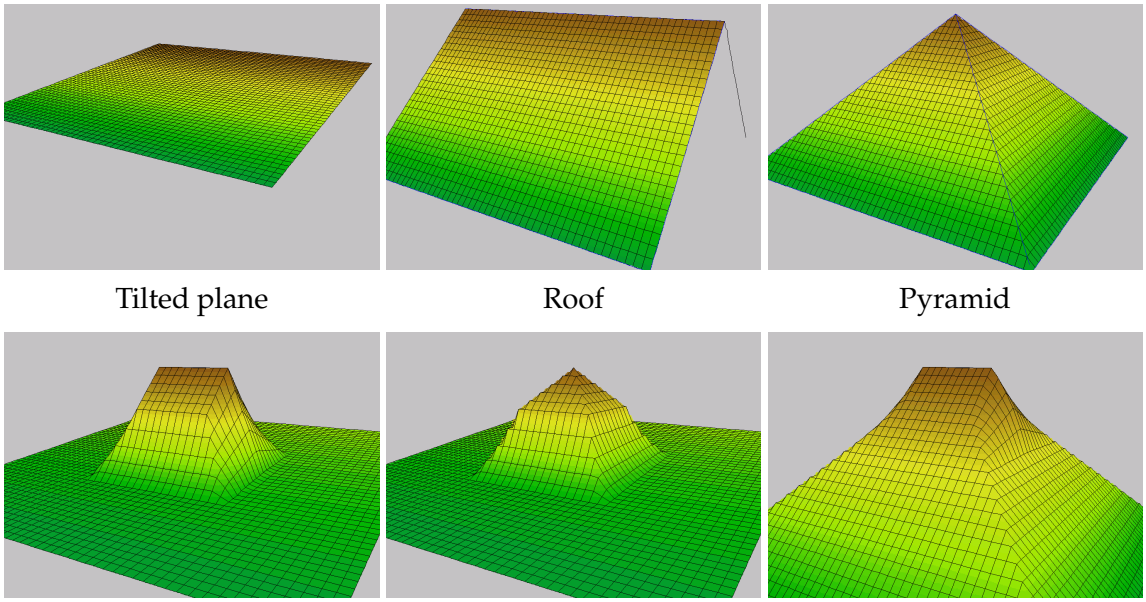
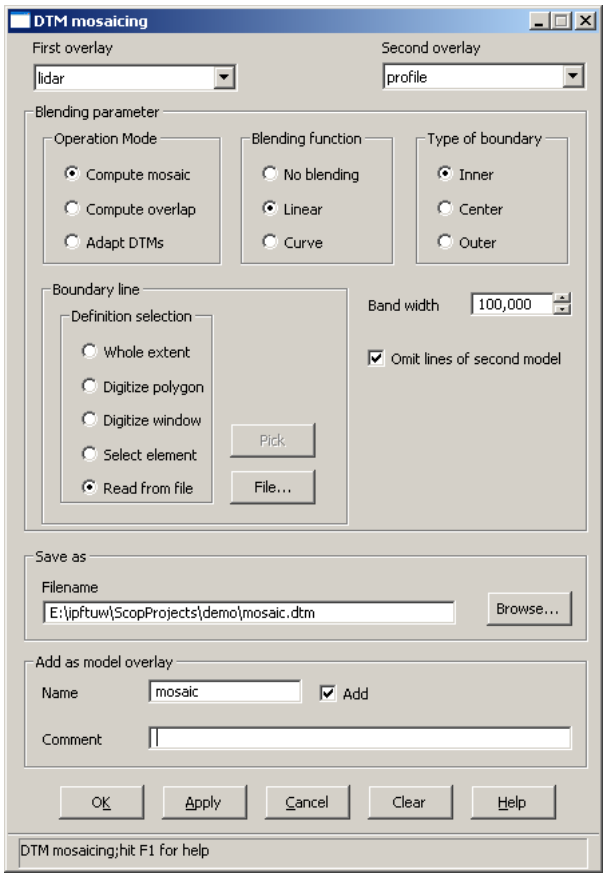


Figure 9.13.: DTM mosaicing examples of synthetic surfaces



- *Compute mosaic:* This is the default mode. The two given DTMs are merged together into one overall DTM.
- *Compute overlap:* In certain cases only the tolerance band is of interest. Above all this becomes true for DTM management in topographic databases, where the accuracies of the different DTMs may serve as a selection criterion.
- *Adapt DTMs:* In this mode the existing DTMs are adapted (corrected) rather than a new DTM file is created. First of all this feature may be used to homogenize the heights at the edge of adjacent DTM tiles.

As described in the above section the transit from one DTM to the other can be controlled by selecting an appropriate *Blending function* (no blending, linear, curve). The tolerance band (i.e. the area where blending should be applied) is determined by a boundary line, the type of the boundary line and the width of the tolerance band. The *boundary line* can be specified either by digitizing a window or polygon in the SCOP++ main graphics panel, by selecting an existing vector element (e.g. isoline, ...) or by reading from a data file. Besides the proprietary boundary line format (see section 9.5.5 for details) DXF and SCOP Winput files are accepted. The proper *Boundary type* has to be specified telling whether the boundary line is the inner/outer edge of the tolerance band or its center. At last the width of the tolerance band is entered in the numeric field *Band width*. Activate the checkbox *Omit lines of second model* to disregard all lines of the second DTM. This may become necessary to avoid inconsistencies.

In case of operation mode *Compute mosaic* or *Compute overlap* the result of the mosaicing operation is a new DTM in SCOP RDH format. The name of the DTM file can either be specified by entering in the text field *Filename* or by browsing for it. Furthermore the resulting DTM can be overtaken directly as a 'model-only' overlay into the current SCOP++ project. To do this activate the checkbox *Add* and enter the name of overlay in the text field *Name*. Specification of a *Comment* is optional.

The calculation can be triggered by pressing button *Apply* or button *Ok*.

Further remarks: Please note

- that the DTM structure (grid width, size of computing unit, ...) is taken from the *First overlay*. For this reason the specification sequence (first/second overlay) is of crucial importance.
- that the model limits as specified in the limits window are taken into account for the generation of the DTM mosaic.
- that the boundary line must be given as a closed polygon (although perhaps the first DTM is not completely inside the second DTM but the DTMs are adjacent). This is necessary since the mosaicing algorithm relies on an 'inside-outside-decision'.

  Read from file supports the following formats: *.lim (SCOP Limits file), *.xy, *.xyz, *.wnp, *.dxf (cf. section9.5.5).

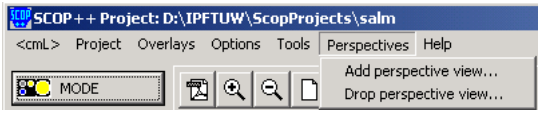
Current restrictions:

- At the moment all line information is disregarded
- Intersections of (multiple) boundary lines is not yet checked.

cmL example

```
@mosaic;
Tools/
  DtmMosaic/
  FirstOverlay=(lidar),    // Assuming there is an ovl. lidar
  SecondOverlay=(profile), // Assuming there is an ovl. profile
  OpMode=(CompMosaic),
  BlendFunc=(Linear),
  TypeBoundary=(Inner),
  Bandwidth=(20),
  OmitLines=(1),
  ReadFromFile,
    File/
    Filename=(d:\data\mergetest\boundary.wnp),
    Ok;
  Add=(1),
  Filename=(d:\data\mergetest\mosaic.dtm),
  Name=(mosaic.dtm),
  Ok;
@endProc;
```

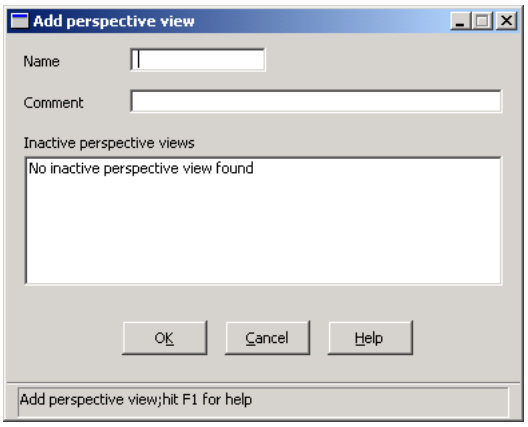
9.6. Perspectives



The bar item *Perspectives* will be locked as long as no project has been opened. The items of this bar item will be locked resp. unlocked logically. For example the drop down dialog *Drop perspective view...* will be locked if there isn't a perspective view to drop.

In each perspective directory the application creates a setup file named *PerspectiveName.stp*. The information concerning the perspective are stored in this file.

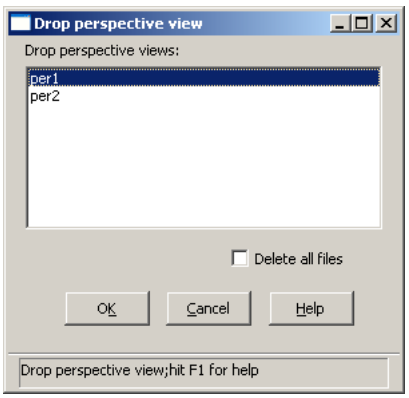
9.6.1. Add Perspective View



This dialog serves for adding perspective views. In the text field *Name* you can enter the name of the perspective view. The second text field *Comment* can be used to provide a few words to comment about perspective view. After clicking on the button *OK* a new perspective environment will be insert into the visualizer. This implies the display of the new button *Perspective View*. Behind this button, situated on the right margin of the main window below the label *Perspective Views*, the window *perspective view* will open.

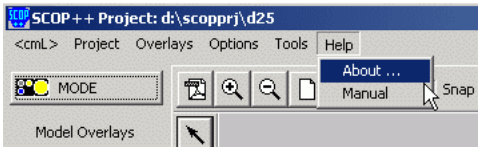
The selection *Inactive perspective views* serves to simplify the task of re-adding perspectives which have been removed. The content of the view is derived from *.stp files, which is located in perspectives directory of the current project. I you select one of the items, the perspective will be inserted again. If there are any inactive perspectives which should appear in the selection, please close and open the dialog and the selection *Inactive perspective views* will be updated.

9.6.2. Drop Perspective View



In the selection *Drop perspective view* you can select the perspective views which should be dropped. Clicking the button *OK* will remove the perspectives from the current project. On response to this action the perspective button on right margin of the main window and all graphics of the adequate perspectives are removed from the screen.
Note also that *Drop perspective view* does not delete any file from the computers hard disc. This includes also the setup file **.stp* holding all the perspective parameters. This may be changed by selecting checkbox *Delete all files*.

9.7. Help



Below the bar item *Help* there is a single window informing about the copyright and the version. Please refer to this version number in case of reporting errors.

Clicking at the bar item *Manual* will start the Acrobat Reader ©and display the manual.

10. Limits

Limits play an essential role in SCOP++. For most of the actions, limits have to be defined. In a SCOP++ project, the definition of limits is centralized in the window *Limits*. The limits defined here are used for all overlays of the current project. For a short overview please refer to section 5.3.13.

The window *Limits* has its own graphics window . It serves as overview window displaying some data of all overlays included in the current project and can be used for quickly digitizing new limits. Data displayed in the graphics window include:

- Model overlays of type both or data-only will display an *overview* of its data, containing all lines (such as break lines, form lines, border lines) and a little percentage of bulk data.
- Model overlays of type model-only extract these lines from the existing DTM-file. If no such lines exist, a rectangle representing the extensions of the overlay will be displayed.
- Image overlays will display their image in reduced resolution. If there are multiple image overlays, these images may hide each other. Raster graphics will not be mixed in the overview graphics panel.

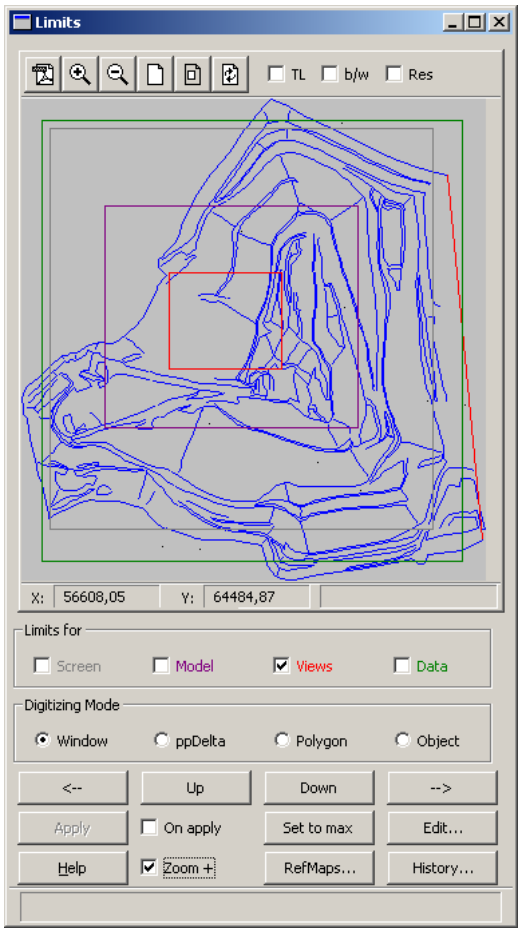


Figure 10.1.: Window LIMITS

Four different categories of limits can be defined. The purposes of these categories are:

- Screen limits: to control the displayed area in the main graphics panel
- Model limits: to delimit the DTM to be interpolated. Note that a DTM can only be derived in areas where data are available. That's why the limits of the resulting DTM may be smaller than the requested ones.
- Views limits: to delimit the extent of *views* to be derived from the DTM. Note, that the views can only be derived where a valid DTM is available. That's why the limits of the resulting views may be smaller than the requested ones.
- Data limits: to delimit the area for *displaying* and *exporting* data.

In the selection *Limits for* below the graphics area the user can chose the category of limits to which the changes he makes will be applied. Please note that the color of the checkboxes in group *Limits for* is identical with the color of the corresponding rectangles in the graphics panel. If two limits are identical, also the color of the checkbox will be identical. The specific color is chosen to somewhat mirror the importance of the limits. According to this,

- one or more limits will be shown in magenta if these contain the model limits,
- they will be shown in light-red if they contain the views limits, but not the model limits,
- they will be shown in green if they contain the data limits but neither the model nor views limits.

The selection *Digitizing mode* states the mode of digitizing. There are four different ways of digitizing. The limits defined by these four modes are always a rectangular area comprising the digitized object:

- Window: to digitize a rectangle whose sides are parallel to the coordinate system
- ppDelta: to digitize a rectangle in arbitrary rotation to the coordinate system
- Polygon: to digitize a polygon from which a rectangle which comprises the whole polygon will be derived (finish digitizing by clicking with the right mouse button)
- Object: to identify an object whose rectangular extents will determine the limits

Several actions editing limits are applied to all categories selected.

The row of buttons below serves for shifting the current limits. The button *Up* will shift them up, the button *Down* down, the button –> right, and the button <– – you guess it – left. The limits will be shifted in a way that a small area of the new rectangle will overlap its ancestor.

The button *Apply* is intended for use in combination which the checkbox *On apply*. Selecting latter will unlock the button *Apply* thus enabling the user to alter the limits without immediate reaction. In this way a iterative limits definition will be possible without frequent and therefore timeconsuming calculations. The new limits are propagated at the very time the user pressed the button *Apply*.

Clicking at button *Set to max* will set the limits to the minima and maxima of all data.

Opening button *Edit...* presents the window *Edit limits* (cf. section 10.1).

On pressing the button *RefMaps...* the reference maps dialog (cf. section 10.2) will appear.

The checkbox *Zoom+* provides a special functionality allowing very detailed zooming and should usually *not* be turned on. This feature is necessary for inspecting graphics results in very high resolution in projects with large extents. The main graphics panel has an upper bound of zooming in. The bound is given by a maximum allowed ratio of displayed to overall area. By selecting the checkbox *Zoom+*, the overall area displayed in the main graphics panel is reduced in order to allow for further zooming in. This functionality is coupled with the Screen limits, i. e. if the "Zoom +" feature is turned on, the displayed area is restricted to three times the screen limits (i.,e. the screen limits enlarged by the same window extent to the left, right, up, and down). Note that graphics contents outside this window are not accessible any more in mode "Zoom +". Thus, do not forget to *switch this feature off again* when you do not need it any more.

Behind the button *History*, the last editing actions can be undone (cf. section 10.3).

10.1. The window *Edit Limits*

Here, the *limits* specified most recently are presented numerically and can be edited.

There is also the possibility to read in limits from a file via the button *Load* and to save them to a file via the button *Save*. The format for the limits to load are the corners of the surrounding polygon, where all polygons loaded will be expanded to a rectangular area containing the polygon. A minimum of two points (four coordinates) are necessary. Saving the limits will yield a file containing two points of the limits rectangle with their x- and y-coordinates.

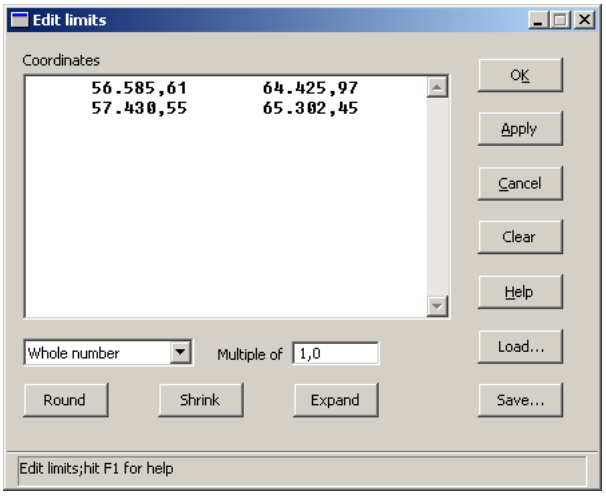


Figure 10.2.: Limits editor

On the bottom of this window a rounding facility is provided. After pressing the button *Round* the limit values will be rounded to the digit which is specified in the selection besides or to a multiple of the value specified with numeric field *Multiple of*. button *Shrink* will round to the next values laying inside the original rectangle, while button *Expand* will round to the next values laying outside the original rectangle. Pay attention that rounding may result in an invalid rectangle, if the number to round is to high. In this case the rounding will be ignored.

Note: When specifying limits by commmand line or command file numeric coordinates have to be specified according to the regional settings from Windows for the decimal symbol. When reading limits from file the decimal dot is used independent from regional settings.

10.2. The window *Limits Reference Maps*

This window is at first sight a list of all displayed reference maps, but it also serves for adding additional reference maps or removing reference maps. You can remove reference maps from the limits graphics panel simply be deselecting the corresponding item in the reference maps dialog. Selecting the white item will be bring the reference to sight again. For selecting and deselecting multiple elements, the Control and Shift keys of your keyboard have to be used.

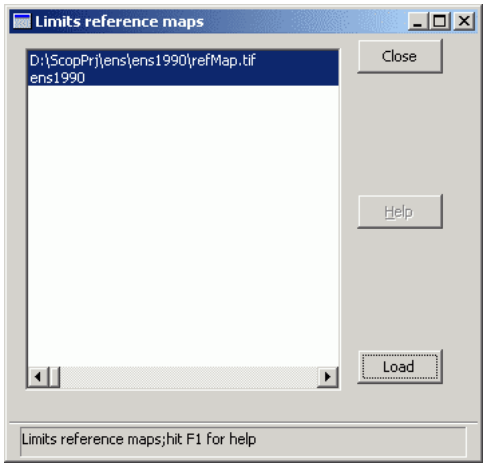


Figure 10.3.: Reference maps window

You can load an additional reference map via the button *Load*. In the appearing file browser you can browse for the file and on clicking the button *Ok* the reference map will be displayed in the limits graphic area. Please note that only following file formats are supported (and that they must have the proper extensions):

- WINPUT with the extensions *.inp and *.wnp
- TIFF with the extension *.tif
- JPEG files with the extension *.jpg
- ZWIFI files with the extension *.zwi

10.3. The window *Limits History*

The history dialog records all user actions and stores the limits corresponding to each of these user action steps. By selecting any of the recorded steps, the corresponding limits are restored. The selected step is moved to the topmost position in the list of recorded user actions.

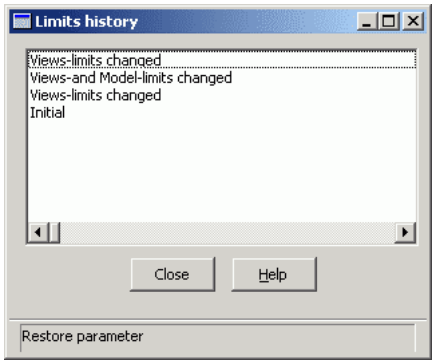


Figure 10.4.: History window

11. Frame

This window (see figure 11.1) serves for manipulation and definition of the parameters to create a frame around the contents in the main graphics panel. This frame will be the same as produced in a plotfile of the `SCOP V3.5` modules.

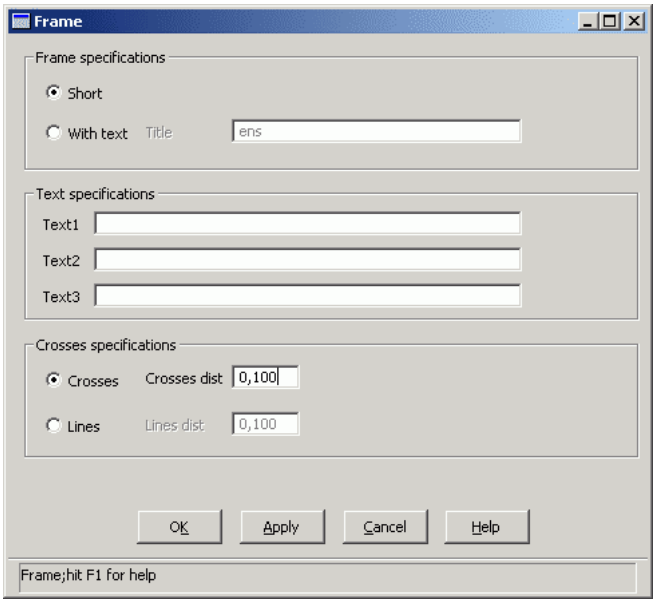


Figure 11.1.: Frame window

As scale of the frame, which will appear in the plot in the right upper corner, the project scale will be used (see section 9.2.8). If the scale is changed and the state-switch *Frame* is in state display, a redisplay of the frame with a new scale will be triggered.

In the group *Frame specifications* you can determine the appearance of the frame. If the radio button *Short* is selected a simple frame will be produced, if the radio button *With text* is selected, a frame with a title will be produced. The title can be entered in the text field *Title*.

In the right lower corner three additional texlines can be placed. The specifications for these can be made in the group *Text specifications* (text field *Text1* to text field *Text3*). Simply type in the desired lines. Mind that the length of this lines is limited with 40 characters. If one of the text fields is left empty this implies that the concerning line won't appear.

In the group *Crosses specifications* the appearance of crosses and lines can be determined. The selection of crosses resp. lines is exclusive. Either one of both possibilities has to be selected. In the adjoined numeric fields the distance for the crosses/lines can be determined.

A commandline / commandfile example for this window is:

cmL example

```
frame/,
  WithText,
  Title = (Epoch 2000),
  text1 = (Test example),
  text2 = (No Name Corporation),
  text2 = (April 2002),
  Lines = 1,
  LinesDist = (0.02),
  OK;
frame = (d);
```

12. Model Overlays

SCOP++ basically is a system to create, maintain and apply digital models. For this reason model overlays are of crucial importance within SCOP++.

There are three types of model overlays available:

Model-only: pre-existing models such as elevation models, digital slope models, ...

Data-only: digital data to be displayed in combination with other model or image overlays

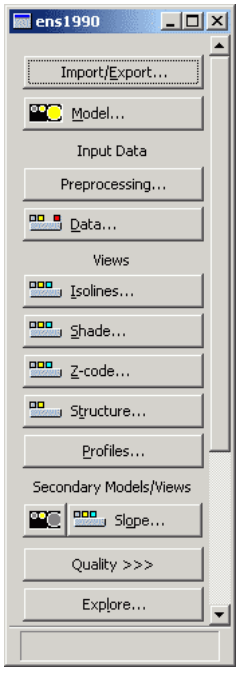
Both: digital data from which models are to be derived

Model-only overlays are for existing digital Models saved as files on the disk. They can be imported to the project so to derive different products (Isolines, ...) from them or to be combined with other digital models to create a new model (difference models, ...).

Data-only overlays consist of digital vector-type data. No models can be derived from data-only overlays. Their contents can be displayed in combination with any other overlay.

Overlays of type "Both" carry digital vector-type data from which models can be derived. Data as well as parameters for model interpolation can be manipulated. Changes in data as well as of interpolation parameters (grid size, limits, ...) will result in automatic re-interpolation of the model – provided that conditions according to the principle of *Lazy Processing* allow for this (see section 5.3.1).

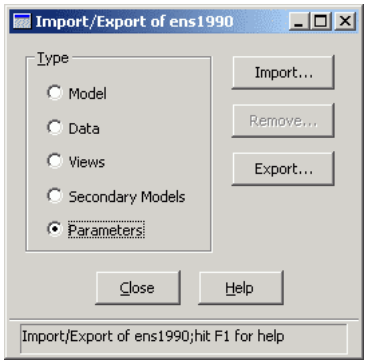
12.1. The window *Model Overlay*



This window can be accessed by clicking at one of the buttons below the at the left side of the main graphics panel each of them corresponding to a certain model overlay.

At the top of this window are the button *Import/Export...* (see section 12.2) and the button *Model* (see section 12.3). Below of them there are two groups of buttons headed by *Input Data* (see section 12.5) and *Views* (see sections 12.6, 12.7, 12.8, 12.9, 12.10).

12.2. Import/Export



This section describes how to

- import data and models from disk files to a model overlay,
- remove data from a model overlay,
- export data, models and views to disk files

Import and export can be accessed by clicking the button *Import/Export...* of a model overlay.

Selecting an item from the selection *Type* means, that data of this category can be handled in the dialogs of the sub-windows. Pressing the buttons *Import..., Remove... and Export...* invokes the appropriate dialog whereas selecting the button *Close* will close the window without any further actions.

A commandline/commandfile example for invoking the import of data dialog is:

cmL example

```
ens2000/
  ImpExp/
    Type=(Data)
    Import/
```

12.2.1. Import Data

In the following we denote with the term *data* primary vector-type data (usually digital elevation data). For storage of data within SCOP++ a geo-coded relational database is used (see section 5.3.9. For that reason it is necessary to import data, stored on external data files located anywhere on the computers periphery (hard-disk, CD-ROM, ...), to the model overlay. Once the data are imported, they can either be displayed and/or edited on the main graphics panel (see section 12.5) or they serve as source for the interpolation of models (see section 12.3).

To open the Import dialog window choose *Data* from the selection *Type* within the Import/Export window and press the button *Import...*.

To import data, enter the name of the file containing digital data in the text field *File* or browse for it. Select the appropriate data format from the list of available data formats and select a code conversion table from the selection *Conversion Table*. Conversion tables are to translate the coding information from the data format specific form (eg. DXF: Layer, Entity) to the common representation (Featurecode, Objecttype) in the database (see sections 5.3.10 and 9.4.3 for details). To use one of the predefined conversion tables choose *STANDARD*. Optionally the (estimated) mean accuracy of the data can be entered in the numeric field *Accuracy*. Clicking the button *OK* or the button *Apply* will start the import process. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window .

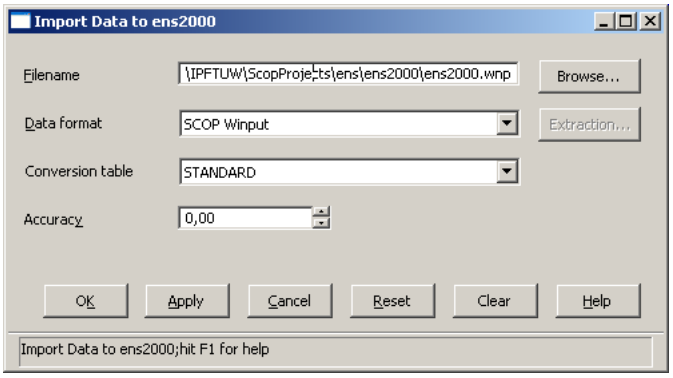


Figure 12.1.: The window *Import Data to overlay*

Please note that an appropriate file extension will be appended if the file name has been entered without extension (ie. .wnp for WINPUT files, etc.). The file extension will be adapted automatically in response to a change of the data format. This automatism is applied as long as no file extension is specified by the user. Note also that specifying a file type in the the file browser does not influence the setting of the data format.

cmL example

```
ens2000/
  ImpExp/
    Type=(Data),
    Import/
      Clear,
      File=(ens2000.dxf),
      Dataformat=("AutoCAD DXF"),
      ConvTable=(STANDARD),
      OK;
    Close;
```

If a data file was successfully imported a log text containing some statistical information is written to the protocol file.

Note that data files can be located anywhere on the disk, not necessarily in the model overlay directory. If the filename is entered without path, then the path to the overlay directory is added automatically. The names of the data files have to be unique within one overlay. Import of Data is not available for model-only overlays.

The following data formats are supported for data import:

SCOP Winput: The proprietary SCOP input data format,

Binary SCOP Winput: Binary SCOP input data format,

AutoCAD DXF: AutoDesks Drawing Interchange File Format,

ArcInfo Generate: Interchange File Format for vector data from ESRI,

XYZ: Text file with 3 coordinates per point in a line,

Binary XYZ: Binary file with 3 double values per point,

ASCII Text File: ASCII text file where columns may be selected to get 3 coordinates (x-y-z) per point,

Kotenband: Interchange File Format for vector data of DGM software TOPSY,

LAS: ASPRS Lidar Data Exchange Format, and

ESRI Shapefile: ESRI Shapefile Format.

For each data format a predefined conversion table is available. As described above, those tables can be accessed by choosing *STANDARD* from the selection *Conversion tables*. The contents of the predefined conversion tables are listed in table 12.1, 12.3, 12.2, 12.5, 12.6, and 12.4.

Import of Data from an ASCII Text File

This format allows the import of XYZ data from a file where other information is included within a line of data. If a filename is selected and this format has been chosen then the button *Extraction...* will be accessible. Clicking at this button a window like in figure 12.2 will open where parameters to select the three relevant columns can be specified.

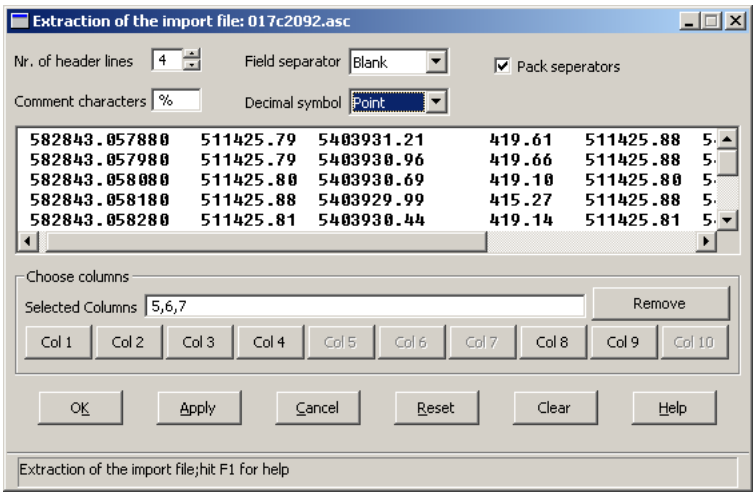


Figure 12.2.: Extraction window for ASCII text file format

Some file formats start with a number of lines containing metadata or other header information. With numeric field *Nr. of header lines* those lines at the beginning of the file may be specified to be excluded from the import process.

Some files may contain comment lines. Usually at the beginning of such lines there are one or two special characters indicating that this line should be treated as comment. With text field *Comment characters* up to two characters may be specified which mark a line as comment.

All other lines in a file should be treated as valid data lines. A valid line will be separated into columns or fields by field separators. The character which is to be used for that purpose can be specified by selection *Field separator*. The available characters are: *blank, tab, comma, semicolon, colon, slash, backslash,* and *stroke*. Leading blanks will be removed from the line before counting the fields. If checkbox *Pack separators* is marked then a series of consecutive field separators will be treated as one. In this mode leading separators will be removed from the line too before cutting the line into fields. In this way data files

with fixed width columns can be read. If checkbox *Pack separators* is not marked then consecutive field separators will represent empty fields. In this way files in CSV (*comma separated values*) format can be read.

With checkbox *Decimal symbol* the character to be used for separating the fractional part of a number from the integer part can be specified.

In the middle of the window there is a text window showing the first few valid lines of the file to be imported. Below of it there is the group *Choose columns* with a series of buttons by which means the available columns can be selected. The specified columns will be inserted to the text field *Selected columns*. The selection process may be restarted by clicking at the button *Remove* which will clear all previously selected columns.

Clicking the button *OK* or the button *Apply* will establish the current values as valid for the import process but will not start it. The import process has to be started from the parent window *Import data to ...*. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window .

Import of Data from a LAS File

LAS (ASPRS Lidar data exchange format) is a binary file format designed for the exchange of data from (airborne) Laser scanning. LAS features the storage of attributed points. Each point is given a return number and the number of all returns originating from the same Laser pulse. Among other attributes, each entry may furthermore specify the point's signal intensity, scan angle, GPS time, and classification. Moreover, the flags *synthetic*, *keypoint*, and *withheld* are associated with each point. As a description of the whole data set, the file header contains information about the file's origin and statistics of the contained point cloud. The file header is supplemented by at least one so-called 'variable length record' that provides geo-coding information. The definition of further records may be proprietary and undisclosed and therefore not be interpretable by Scop++. Scop++ supports the LAS file format up to version 1.2. Detailed information on LAS is provided at www.lasformat.org.

Selecting LAS as file format activates the button *Extraction...*, which grants access to the window *Echo extraction of the import file* shown in figure 12.3. This dialog in turn offers an insight into LAS files and allows for the definition of LAS import options.

The window's upper part permits an insight into the selected file(s) on two levels. Having activated radio button *Read header*, hitting the button *Inspect* yields a fast dump of the LAS file header into the text box below. However, the file header merely provides humble information about the data contents, which may furthermore be incorrect due to deficiencies in the software that generated the file. For this reason, Scop++ offers a deeper insight into the file contents through the radio button *Analyse data*. Having hit button *Inspect*, the file is read through and data statistics are accumulated, which may take a little while. Finally, the number of *First*, *Last*, and *All* echoes, the number of points regarding classification and the point data flags, and the range of point intensities and scan angles is output. Depending on the point record format, also the range of GPS times may be given. In any case, the file header is provided with content corrected according to the point data.

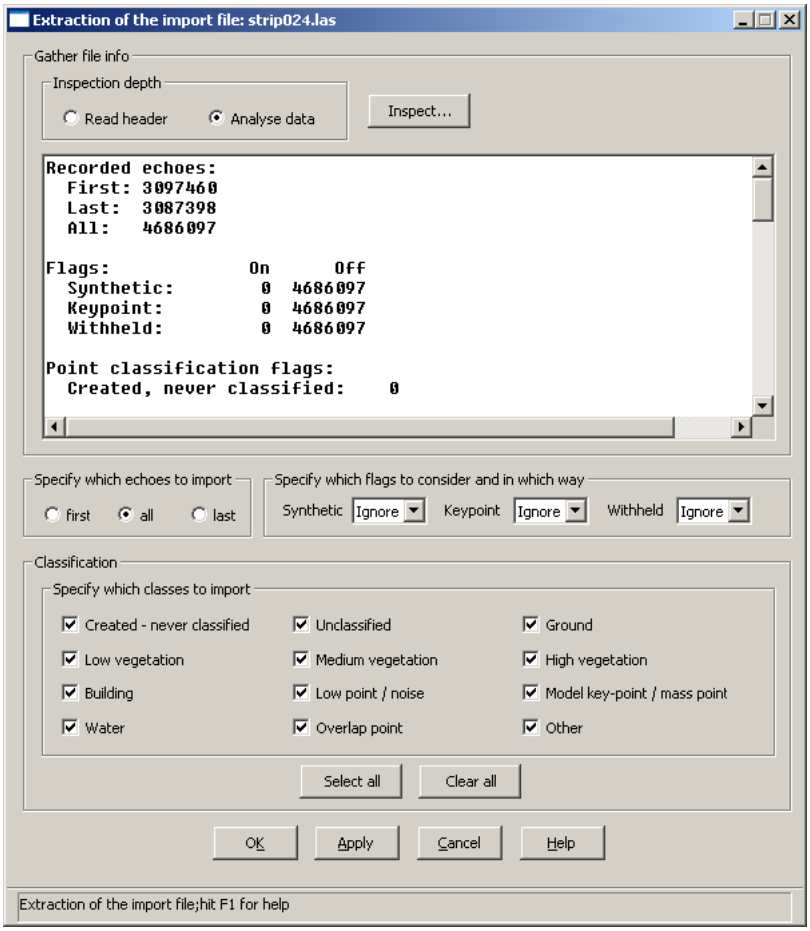


Figure 12.3.: Extraction window for LAS file format

The left side of the middle part of this window permits the definition of the Laser echoes to import: either *First*, *All*, or *Last*. *First* echoes are defined as LAS points having a return number of zero or one, while *Last* echo points feature identical return numbers and numbers of returns. Kindly note that generally, the subsets of *First* and *Last* points will overlap. Selecting *All* naturally results in the import of all points, regardless of the return number.

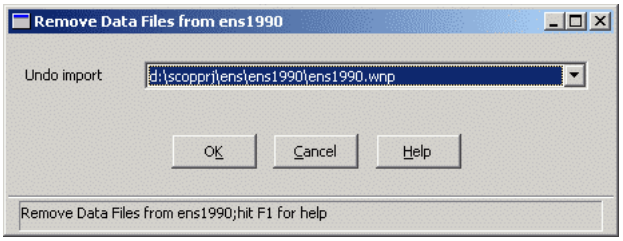
The right side of the central part of this window allows for restricting the import of data according to the point flags *synthetic*, *keypoint*, and *withheld*. In the selection of which points to import from file, each of these flags may either be ignored (*Ignore*), or be required to be set (*On*) or not (*Off*). For a point to be imported, all applied restrictions must hold e.g. if *synthetic* is *On*, *keypoint* is ignored, and *withheld* is *Off*, then only those points will be imported whose *synthetic*-bit is 1 and whose *withheld*-bit is 0, irrespective of the value of *keypoint*.

Finally, the lower part of this window allows for filtering the data to import by classification number: for each class defined by LAS, an according check box is provided. Points featuring a classification number that corresponds to an active check box will be imported, while those related to inactive boxes will not. While LAS uses 5 bits to store classifications, resulting in 32 possible values, LAS (currently) only defines 11 of them.

Each of these 11 defined classifications may be selected for import separately. Check box *Other* allows for importing data holding any other, undefined classifications.

For a point to be imported, the restrictions from all three groups (echo, flag, classification) must be fulfilled.

12.2.2. Remove Data



Data files imported to a model overlay are stored in the project database, from where they can also be removed again. Note that data files are not deleted from the computers hard disc during this process, but their contents are removed from the project table.

To open the Remove dialog window choose *Data* from the selection *Type* within the Import/Export window and press the button *Remove...*.

To remove data (undo import), select the name of the data file from the list of imported data files and click the button *OK*. The contents of the data file will be removed from the overlay as well as from the database. Pressing the button *Cancel* will close the window without removing any data.

cmL example

```
ens2000/
  ImpExp/
    Type=(Data),
    Remove/
      UndoImport=("d:\scop_projects\ens\ens2000\ens2000.wnp"),
      OK;
    Close;
```

Note that the Remove Data Dialog is not available for model-only overlays.

12.2.3. Export Data

This section describes how the data of a model overlay can be exported to a disk file. This feature is useful to transfer data to other systems (eg. CAD, GIS, ...).

To open the Export dialog window choose *Data* from the selection *Type* within the Import/Export window and press the button *Export...*.

The dialog window *Export Data of overlay* contains the text field *File* to enter the name of a disk file, the selection *Data Format* to select the appropriate data format and the selection *Conversion Table* to define the Conversion Table for translation of the coding information. The filename can also be selected using the file browser . Concerning Conversion Tables the same rules apply as for data import (see section 12.2.1. Clicking the button *OK* or the button *Apply* will start the export process. Pressing the button *Cancel* or the button *Reset*

Table 12.1.: Standard Code Conversion Table for WINPUT and Binary WINPUT

FEATURECODE	OBJECTTYPE	WNP-CODE
PROFILE	CLUSTER	10
PROFILE	CLUSTER	11
GRIDPOINT	CLUSTER	12
ALIGNEMENT	LINE	15
CROSS_SECTION	LINE	16
CONTOURLINE	LINE	20
CONTOURLINE	LINE	21
RANDOM	CLUSTER	30
SPOTHEIGHT_HIGH	SYMBOL	31
SPOTHEIGHT	SYMBOL	31
SPOTHEIGHT_LOW	SYMBOL	32
FORMLINE	LINE	40
FORMLINE	AREA	41
BREAKLINE	LINE	50
BREAKLINE	AREA	51
BREAK_BORDER_OMIT_RIGHT	LINE	52
BREAK_BORDER_OMIT_RIGHT	AREA	53
BREAK_BORDER_OMIT_LEFT	LINE	54
BREAK_BORDER_OMIT_LEFT	AREA	55
BORDERLINE_OMIT_RIGHT	LINE	60
BORDERLINE_OMIT_RIGHT	AREA	61
BORDERLINE_OMIT_RIGHT_NOZ	LINE	62
BORDERLINE_OMIT_RIGHT_NOZ	AREA	63
BORDERLINE_OMIT_LEFT	LINE	64
BORDERLINE_OMIT_LEFT	AREA	65
BORDERLINE_OMIT_LEFT_NOZ	LINE	66
BORDERLINE_OMIT_LEFT_NOZ	AREA	67
BORDERLINE	AREA	68
EXCLUSIONLINE	AREA	69
SINGULAR	POINT	70
SITUATION	LINE	80
SITUATION1	LINE	81
SITUATION2	LINE	82
SITUATION3	LINE	83
SITUATION4	LINE	84
SITUATION5	LINE	85
SITUATION6	LINE	86
ObjectShape	LINE	87
SITUATION8	LINE	88
SITUATION9	LINE	89

Table 12.2.: Standard Code Conversion Table for ArcInfo Generate

FEATURECODE	OBJECTTYPE	SFT
RANDOM	CLUSTER	1
RANDOM	LINE	1
PROFILE	CLUSTER	1
GRIDPOINT	CLUSTER	1
ALIGNEMENT	LINE	1
CROSS_SECTION	LINE	1
CONTOURLINE	LINE	2
SPOTHEIGHT_HIGH	SYMBOL	1
SPOTHEIGHT	SYMBOL	1
SPOTHEIGHT_LOW	SYMBOL	1
FORMLINE	LINE	2
FORMLINE	AREA	2
BREAKLINE	LINE	3
BREAKLINE	AREA	3
BREAK_BORDER_OMIT_RIGHT	LINE	7
BREAK_BORDER_OMIT_RIGHT	AREA	7
BREAK_BORDER_OMIT_LEFT	LINE	7
BREAK_BORDER_OMIT_LEFT	AREA	7
BORDERLINE_OMIT_RIGHT	LINE	9
BORDERLINE_OMIT_RIGHT	AREA	9
BORDERLINE_OMIT_RIGHT_NOZ	LINE	9
BORDERLINE_OMIT_RIGHT_NOZ	AREA	9
BORDERLINE_OMIT_LEFT	LINE	9
BORDERLINE_OMIT_LEFT	AREA	9
BORDERLINE_OMIT_LEFT_NOZ	LINE	9
BORDERLINE_OMIT_LEFT_NOZ	AREA	9
BORDERLINE	AREA	9
EXCLUSIONLINE	AREA	9
SITUATION	LINE	0
SITUATION1	LINE	0
SITUATION2	LINE	0
SITUATION3	LINE	0
SITUATION4	LINE	0
SITUATION5	LINE	0
SITUATION6	LINE	0
ObjectShape	LINE	0
SITUATION8	LINE	0
SITUATION9	LINE	0
PLANIMETRY	POINT	0
PLANIMETRY	LINE	0
SINGULAR	LINE	0

Table 12.3.: Standard Code Conversion Table for DXF

FEATURECODE	OBJECTTYPE	LAYER	ENTITY
PROFILE	CLUSTER	PROFILE	POINT
GRIDPOINT	CLUSTER	GRIDPOINT	POINT
ALIGNEMENT	LINE)	ALIGNMENT	POLYLINE
CROSS_SECTION	LINE	CROSSSEC	POLYLINE
CONTOURLINE	LINE	CONTOUR	POLYLINE
RANDOM	CLUSTER	POINTS	POINT
SPOTHEIGHT_HIGH	SYMBOL	SPOTHT	POINT
SPOTHEIGHT	SYMBOL	SPOTHT	POINT
SPOTHEIGHT_LOW	SYMBOL	SPOTLO	POINT
FORMLINE	LINE	FORMLIN	POLYLINE
FORMLINE	AREA	FORMLIN	POLYLINE
BREAKLINE	LINE	BREAKLIN	POLYLINE
BREAKLINE	AREA	BREAKLIN	POLYLINE
BREAK_BORDER_OMIT_RIGHT	LINE	BRBOLINR	POLYLINE
BREAK_BORDER_OMIT_RIGHT	AREA	BRBOLINR	POLYLINE
BREAK_BORDER_OMIT_LEFT	LINE	BRBOLINL	POLYLINE
BREAK_BORDER_OMIT_LEFT	AREA	BRBOLINL	POLYLINE
BORDERLINE_OMIT_RIGHT	LINE	BORDLINR	POLYLINE
BORDERLINE_OMIT_RIGHT	AREA	BORDLINR	POLYLINE
BORDERLINE_OMIT_RIGHT_NOZ	LINE	BORDNOZR	POLYLINE
BORDERLINE_OMIT_RIGHT_NOZ	AREA	BORDNOZR	POLYLINE
BORDERLINE_OMIT_LEFT	LINE	BORDLINL	POLYLINE
BORDERLINE_OMIT_LEFT	AREA	BORDLINL	POLYLINE
BORDERLINE_OMIT_LEFT_NOZ	LINE	BORDNOZL	POLYLINE
BORDERLINE_OMIT_LEFT_NOZ	AREA	BORDNOZL	POLYLINE
BORDERLINE	AREA	BORDERLI	POLYLINE
EXCLUSIONLINE	AREA	EXCLUDELI	POLYLINE
SINGULAR	POINT	SINGPOIN	POINT
SITUATION	LINE	SITULIN	POLYLINE
SITUATION1	LINE	SITULIN1	POLYLINE
SITUATION2	LINE	SITULIN2	POLYLINE
SITUATION3	LINE	SITULIN3	POLYLINE
SITUATION4	LINE	SITULIN4	POLYLINE
SITUATION5	LINE	SITULIN5	POLYLINE
SITUATION6	LINE	SITULIN6	POLYLINE
ObjectShape	LINE	SITULIN7	POLYLINE
SITUATION8	LINE	SITULIN8	POLYLINE
SITUATION9	LINE	SITULIN9	POLYLINE
CONTOURLINE	LINE	PLINE001	POLYLINE
CONTOURLINE	LINE	PLINE002	POLYLINE
CONTOURLINE	LINE	PLINE003	POLYLINE
CONTOURLINE	LINE	PLINE004	POLYLINE
CONTOURLINE	LINE	PLINE005	POLYLINE
CONTOURLINE	LINE	PLINE006	POLYLINE
CONTOURLINE	LINE	PLINE007	POLYLINE
CONTOURLINE	LINE	PLINE008	POLYLINE
CONTOURLINE	LINE	PLINE009	POLYLINE
CONTOURLINE	LINE	LINE001	POLYLINE
CONTOURLINE	LINE	LINE002	POLYLINE
CONTOURLINE	LINE	LINE003	POLYLINE
CONTOURLINE	LINE	LINE004	POLYLINE

Table 12.4.: Standard Code Conversion Table for ESRI Shapefile

FEATURECODE	OBJECTTYPE	SHAPETYPE
RANDOM	CLUSTER	MultiPointZ
GRIDPOINT	CLUSTER	MultiPointZ
BREAKLINE	LINE	PolyLineZ
BREAKLINE	AREA	PolyLineZ
FORMLINE	LINE	PolyLineZ
FORMLINE	AREA	PolyLineZ
BREAK_BORDER_OMIT_RIGHT	LINE	PolyLineZ
BREAK_BORDER_OMIT_RIGHT	AREA	PolyLineZ
BREAK_BORDER_OMIT_LEFT	LINE	PolyLineZ
BREAK_BORDER_OMIT_LEFT	AREA	PolyLineZ
BORDERLINE_OMIT_RIGHT	AREA	PolygonZ
BORDERLINE_OMIT_RIGHT	LINE	PolygonZ
BORDERLINE_OMIT_RIGHT_NOZ	AREA	Polygon
BORDERLINE_OMIT_RIGHT_NOZ	LINE	Polygon
BORDERLINE_OMIT_LEFT	AREA	PolygonZ
BORDERLINE_OMIT_LEFT	LINE	PolygonZ
BORDERLINE_OMIT_LEFT_NOZ	AREA	Polygon
BORDERLINE_OMIT_LEFT_NOZ	LINE	Polygon
BORDERLINE	AREA	Polygon
EXCLUSIONLINE	AREA	Polygon
SPOTHEIGHT_HIGH	SYMBOL	MultiPointZ
SPOTHEIGHT	SYMBOL	MultiPointZ
SPOTHEIGHT_LOW	SYMBOL	MultiPointZ
SPOTHEIGHT_HIGH	SYMBOL	PointZ
SPOTHEIGHT	SYMBOL	PointZ
SPOTHEIGHT_LOW	SYMBOL	PointZ
PROFILE	CLUSTER	MultiPointZ
ALIGNEMENT	LINE	PolyLineZ
CROSS_SECTION	LINE	PolyLineZ
CONTOURLINE	LINE	PolyLineZ
SITUATION	LINE	PolyLineZ
SITUATION1	LINE	PolyLineZ
SITUATION2	LINE	PolyLineZ
SITUATION3	LINE	PolyLineZ
SITUATION4	LINE	PolyLineZ
SITUATION5	LINE	PolyLineZ
SITUATION6	LINE	PolyLineZ
ObjectShape	LINE	PolygonZ
SITUATION8	LINE	PolyLineZ
SITUATION9	LINE	PolyLineZ
PLANIMETRY	POINT	MultiPoint
PLANIMETRY	LINE	PolyLine
SINGULAR	POINT	MultiPointZ

Table 12.5.: Standard Code Conversion Table for XYZ, Binary XYZ and ASCII Text File

FEATURECODE	OBJECTTYPE
RANDOM	CLUSTER
PROFILE	CLUSTER
GRIDPOINT	CLUSTER
ALIGNEMENT	LINE
CROSS_SECTION	LINE
CONTOURLINE	LINE
RANDOM	CLUSTER
SPOTHEIGHT_HIGH	SYMBOL
SPOTHEIGHT	SYMBOL
SPOTHEIGHT_LOW	SYMBOL
FORMLINE	LINE
FORMLINE	AREA
BREAKLINE	LINE
BREAKLINE	AREA
BREAK_BORDER_OMIT_RIGHT	LINE
BREAK_BORDER_OMIT_RIGHT	AREA
BREAK_BORDER_OMIT_LEFT	LINE
BREAK_BORDER_OMIT_LEFT	AREA
BORDERLINE_OMIT_RIGHT	LINE
BORDERLINE_OMIT_RIGHT	AREA
BORDERLINE_OMIT_RIGHT_NOZ	LINE
BORDERLINE_OMIT_RIGHT_NOZ	AREA
BORDERLINE_OMIT_LEFT	LINE
BORDERLINE_OMIT_LEFT	AREA
BORDERLINE_OMIT_LEFT_NOZ	LINE
BORDERLINE_OMIT_LEFT_NOZ	AREA
BORDERLINE	AREA
EXCLUSIONLINE	AREA
SINGULAR	POINT
SITUATION	LINE
SITUATION1	LINE
SITUATION2	LINE
SITUATION3	LINE
SITUATION4	LINE
SITUATION5	LINE
SITUATION6	LINE
ObjectShape	LINE
SITUATION8	LINE
SITUATION9	LINE
PLANIMETRY	POINT
PLANIMETRY	LINE

Table 12.6.: Standard Code Conversion Table for Kotenband

FEATURECODE	OBJECTTYPE	KEYCODE
Gelaendepunkt	CLUSTER	0
Gelaendepunkt	CLUSTER	1
Gelaendepunkt	CLUSTER	2
Wegepunkt	POINT	20
Wegepunkt	POINT	21
Markanter_Punkt	SYMBOL	30
Markanter_Punkt	SYMBOL	31
Aussparungsflaeche_K	AREA	51
Geripplinie	LINE	61
Aussparungsflaeche_oH	AREA	71
Gelaendekante	LINE	81
Aussparungsflaeche	AREA	91
Gelaendepunkt_Q2	CLUSTER	200
Geripplinie_Q2	LINE	261
Gelaendekante_Q2	LINE	281

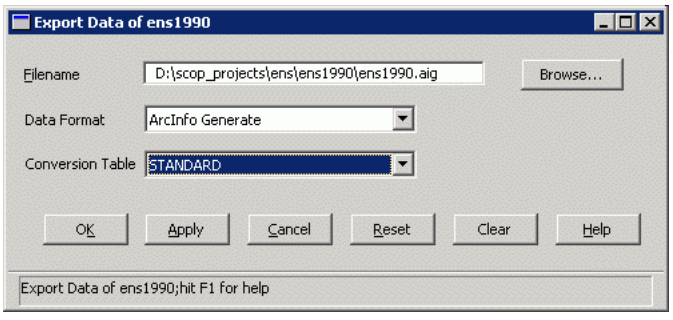


Figure 12.4.: The window *Export Data of overlay*

will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window .

Please note that an appropriate file extension will be appended if the file name has been entered without extension (ie. .dxf for AutoCAD DXF files, etc.). The file extension will be adapted automatically in response to a change of the data format. This automatism is applied as long as no file extension is specified by the user. Note also that specifying a file type in the the file browser does not influence the setting of the data format.

Data are stored in a single file for all data formats except ArcInfo Generate. The ArcInfo Generate file format provides a different syntax for point and line files. For that reason data are stored in two separate files. SCOP++ automatically appends the extension ".pts" for point files and ".lin" for line files. If there is no line information available in the data set, then data are stored using the original file name entered by the user.

cmL example

```
ens2000/
  ImpExp/
    Type=(Data),
    Export/
      File=("d:\scop_projects\ens\ens2000\ens2000.aig"),
      DataFormat=("ArcInfo Generate"),
      ConvTable=(STANDARD),
      OK;
    Close;
```

Note that if the filename is entered without path, then the path to the overlay directory is added automatically. Export of Data is not available for model-only overlays.

12.2.4. Import Model

This section describes how digital models can be imported to a model overlay. It is available for model-only overlays. Import of models is needed to take of over digital models either from a different SCOP++ installation, from a different SCOP++ project, from the current project or finally from any other data source. There are different applications for taking over a digital model from the current project as a model-only overlay:

- to store the current state of an overlay of type "both". By creating a model-only overlay the model is separated from the data and thus preserved from changes.
- to take over a digital model calculated by means of DTM algebra (difference models, slope models, ...) as a model-only overlay.
- ...

To open the Import dialog window choose *Model* from the selection *Type* within the Import/Export window and press the button *Import...*.

The dialog window *Import Model to overlay* contains the text field *File* to enter the name of a model file and the selection *Data Format* to select the appropriate data format from a list of available data formats. The filename can also be selected using the file browser . Clicking the button *OK* or the button *Apply* will start the import process. Pressing the

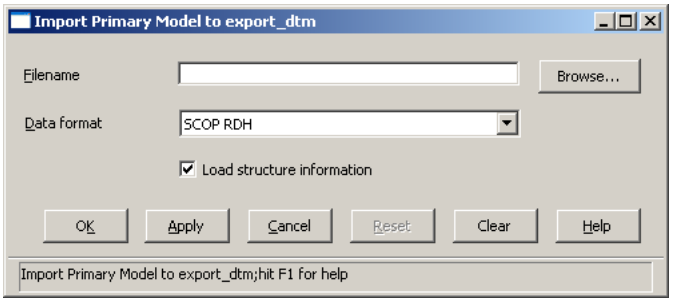


Figure 12.5.: The window *Import Model to overlay*

button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window .

The following data formats are supported for model import:

SCOP RDH: The proprietary SCOP DTM file format,

ArcInfo ASCII Grid: ESRI's ASCII grid file format

Raw Binary BIL (band interleaved by line) file format

TIFF Tagged Image File Format

USGS DEM Raster

USGS SDTS Raster

SRTM Shuttle Radar Topography Mission

DTED Level 0,1,2 DTED Elevation Raster

ESRI .hdr Labelled Raster

ENVI .hdr Labelled Raster

ERMapper

Golden Software ASCII Grid

Golden Software Binary Grid

ERDAS Imagine

Intergraph Raster

NASA Planetary Data System

cmL example

```
diffModel/
  ImpExp/
    Type=(Model),
    Import/
      File=(diffModel.dtm),
      DataFormat=("SCOP RDH"),
      OK;
    Close;
```

If checkbox *Load structure information* is selected then the embedded line information will be extracted from the DTM file to make a visualization within the main graphics panel possible.

If a model file was successfully imported a log text containing some statistical information is written to the protocol file.

Note that model files can be located anywhere on the disk, not necessarily in the model overlay directory. If the filename is entered without path, then the path to the overlay directory is added automatically.

12.2.5. Remove Model

This section describes how model files can be removed from a model overlay. It is available for model-only overlays.

To open the Remove dialog window choose *Model* from the selection *Type* within the Import/Export window and press the button *Remove...*.

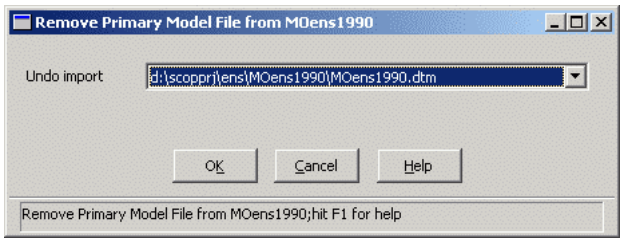


Figure 12.6.: The window *Remove Model from overlay*

To remove a model select the name of the model file from the list of imported model files and click the button *OK*. Pressing the button *Cancel* will close the window without removing any model file.

cmL example

```
diffModel/
  ImpExp/
    Type=(Model),
    Remove/
      RemoveFile=("d:\scop_projects\ens\diffModel\diffModel.dtm"),
      OK;
    Close;
```

12.2.6. Export Model

This section describes how a SCOP++ model can be exported to a disk file. This feature is useful to

- Transfer a SCOP++ model to other systems (eg. CAD, GIS, ...).
- Add a secondary model as the ”primary” of a model-only overlay to the current project.

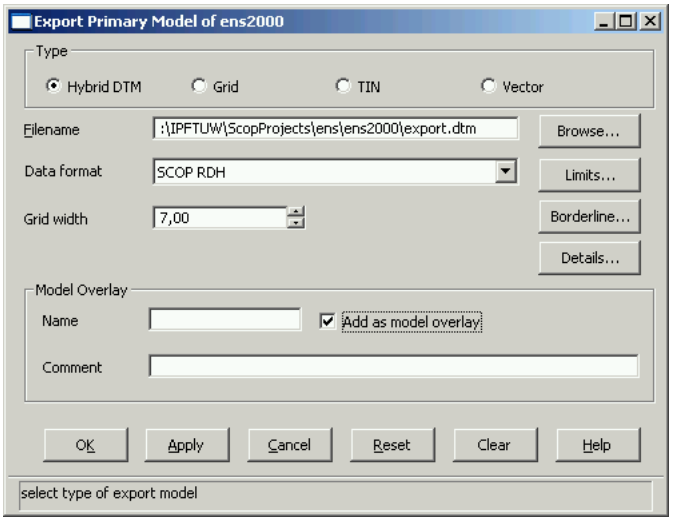


Figure 12.7.: The window *Export Model of overlay*

To open the Export dialog window choose *Model* from the selection *Type* within the Import/Export window and press the button *Export...*.

The proprietary SCOP DTM structure is a hybrid model (5.1) consisting of a regular grid and intermeshed lines (break lines, form lines, border lines) and spot heights (peaks, sinks). The SCOP DTM structure combines the advantage of regular surface description in smooth areas and good geomorphological quality through the embedded structure information. Since the hybrid DTM structure is rarely supported by GIS or CAD program systems, SCOP++ also provides functionality to export the model as a regular grid (raster), Triangulated Irregular Network (TIN) or as discrete points and lines (Vector).

Thus, the dialog window *Export Model of overlay* contains radio buttons *Hybrid DTM, Grid, TIN and Vector* to specify the general data structure in which the model is to be exported. For each type an additional window containing detail parameters for the respective data structure opens simultaneously. Please see below for a detailed description of all these additional parameters. The main export dialog window provides the text field *File* to enter the name of a disk file and the selection *Data Format* to select the appropriate data format from a list of available data formats. The filename can also be selected using the file browser . Please note that an appropriate file extension will be appended if the file name has been entered without extension (ie. .wrl for VRML files, etc.). The file extension will be adapted automatically in response to a change of the data format. This automatism is applied as long as no file extension is specified by the user. Note also that specifying a file type in the the file browser does not influence the setting of the data format.

Furthermore the grid width of the exported model can be specified using the numeric field *Grid width*. The default value for this field is the grid width of the current model. This field is automatically updated when a new model is calculated, as long as a specific value is entered by the user. Please note, that this option is available in SCOP++ Analyzer only.

Optionally the exported model can be inserted to the current project as a model-only overlay. For this purpose the group *Model Overlay* contains the checkbox *Add as Model*

Overlay. Clicking this checkbox the text field *Overlay* and the text field *Comment* get accessible and the name of a model-only overlay and an optional comment can be entered there. Clicking the button *OK* or the button *Apply* will start the export process. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window . In the case the checkbox *Add as Model Overlay* is activated and the text field *Overlay* is not empty, a new model-only overlay will be added to the current project. The exported model file will be imported to this overlay automatically. Please note that adding a model-only overlay is supported for SCOP models only. If the filename is entered without path, then the path to the overlay directory is added automatically. Export of Models is not available for data-only overlays.

Definition of limits for export: In the window beneath the state-switch *Limits...* (see fig 12.8) limits for the export of the model can be defined.

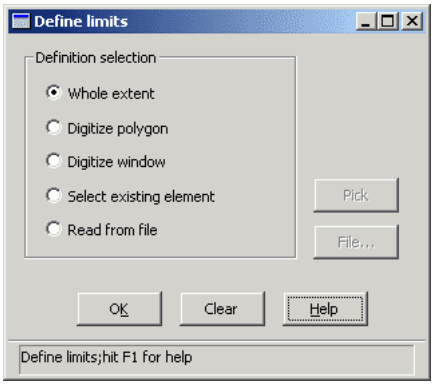


Figure 12.8.: The window *Define Limits* for export of models

The limits can be specified by:

- Whole extent: This is selected by default.
- Digitizing a polygon in the main graphics panel: The polygon can be digitized after pressing the button *Pick*.
- Digitizing a window in the main graphics panel: The window can be digitized after pressing the button *Pick*.
- Selecting an existing item in the main graphics panel: The element can be selected after pressing the button *Pick*.[1]
- Reading a polygon from file: Specify the file in the window beneath the state-switch *File...*. The following file formats are supported:
  - Simple ASCII (one coordinate duple per line, at least 4 coordinates = 2 limit points)
  - SCOP Winput (all files with extension .wnp are regarded as Winput files)
  - AutoCAD DXF (all files with extension .dxf are regarded as DXF files)

[1] If selecting the borderline of the model the result may be unexpected. The borderline may be fragmented because points of the borderline will be classified accidentally as inside or outside. A possible solution is to choose *Omit borderlines* within *Detailed parameters*.

Definition of additional borderlines: In the window beneath the state-switch *Borderlines...* (see fig 12.9) additional borderlines for model export in SCOP RDH file format can be defined. This is particularly useful for model-only overlays, if the original model does not contain a borderline. On the other hand this feature can also be used to exclude/include specific parts of an entire model. The definition of the additional borderlines is done just as described above for the limits. The only difference to the limits specification is that multi-line definition is allowed for borderlines (i.e. there may be more than one borderline). In the case that the borderlines are read from a data file which are not structured according to the SCOP Winput or DXF format specifications, the same file format as for the Volume/Surface classes is expected (see 9.5.5 for more details). Please note that the button *Borderlines...* is only accessible for export in SCOP RDH file format. For digitized borderlines the rotation sense is used to differentiate outer border lines and exclusion lines. Lines oriented counterclockwise are treated as border lines whereas clockwise orientation indicates exclusion lines. Borderlines read from SCOP Winput files are interpreted according to their respective Winput Code. In all other cases (DXF, ASCII) the border lines are strictly handled as outer border lines. Please note that no consistency check is performed. Thus, the user should ensure that the borderlines to be inserted are not contradicting.

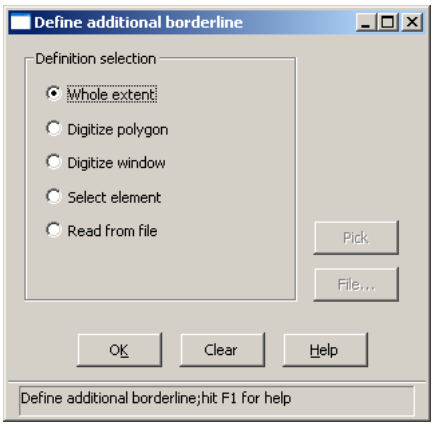


Figure 12.9.: The window *Define additional borderlines* for export of models

Hybrid DTM export

To export the model in the proprietary hybrid structure, select *Hybrid DTM* from the selection *Type* within the main export dialog window . The only available data format in this case is SCOP RDH.

The window *Detail parameters* contains the selection *Contents* allowing to activate or deactivate *Spotheights*, *Formlines*, *Breaklines* and *Borderlines*. This is particularly useful to suppress exclusion areas in the exported SCOP RDH file, in which case the checkbox *Borderlines* needs to be deactivated. Another application is to deactivate all items and thus, to export a regular grid based on the original hybrid model.

The parameters within the group *Extents* allow fine adjustment of the rectangular model area. Activate the checkbox *Snap to grid* to ensure the coincidence of the original and

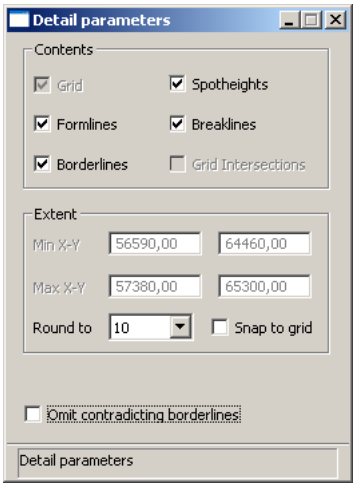


Figure 12.10.: The window *Detail parameters* for Hybrid DTM export

the exported grid if an arbitrary user defined limit polygon was specified (precondition: original grid width = grid width of export file). Alternatively the model extents may be rounded to powers of ten (selection *Round to:*).

Finally the checkbox *Omit contradicting borderlines* can be activated to avoid conflicts between the borderlines of the original model and the user defined borderlines. Of course this parameter only takes effect if additional borderlines are defined in the main export window. In case of contradiction (intersection) the user defined borderlines are favored over the borderlines of the original model.

cmL example

```
ens2000/
  ImpExp/
    Type=(Model),
    Export/
      Clear,
      Type=("Hybrid DTM"),
      File=(dtm2000.dtm),
      DataFormat=("SCOP RDH"),
      Borderline/
        ReadFromFile,
        File/
          Filename=(exclusionline.wnp),
          Ok;
        Ok;
      Details,
        Spotheights=(1),
        Formlines=(1),
        Breaklines=(1),
        Borderlines=(1),
        OmitContradict=(1);
      AddAsModelOverlay=(on),
      Name=(dtm2000),
```

```
    Comment=(Final version of DTM for epoch 2000),
    OK;
  Close;
```

Grid export

To export the model in regular grid (raster) structure select *Grid* from the selection *Type* within the main export dialog window .

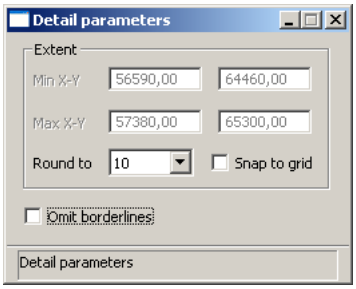


Figure 12.11.: The window *Detail parameters* for Grid export

The following data formats are supported for grid export:

ArcInfo Ascii Grid: ASCII grid format of ESRI

DTED: Digital Terrain Elevation Data, binary

VRML: Virtual Reality Modelling Language, Format to describe 3D-scenes for the internet, ASCII

DGM-Band: DGM file format of program system TOPSY, ASCII
X, Y, Z, Quality, grad in x[gon], grad in y[gon], slope [gon], exposition[gon]

AutoCAD DXF: wire frame based on regular grid, ASCII

Raw Binary: BIL (band interleaved by line) file format

XYZ: simple list of coordinate-triples, ASCII and binary

XYZ-Slope : X, Y, Z, slope [deg], exposition [deg], ASCII

TIFF Tagged Image File Format

USGS DEM Raster

SRTM Shuttle Radar Topography Mission

ESRI .hdr Labelled Raster

ENVI .hdr Labelled Raster

ERMapper

Golden Software ASCII Grid

Golden Software Binary Grid

ERDAS Imagine

Intergraph Raster

Within the window *Detail parameters* the group *Extents* allows fine adjustment of the rectangular model area. Activate the checkbox *Snap to grid* to ensure the coincidence of the original and the exported grid if an arbitrary user defined limit polygon was specified (precondition: original grid width = grid width of export file). Alternatively the model extents may be rounded to powers of ten (selection *Round to:*). The checkbox *Omit borderline* controls the behavior in excluded areas. If activated all grid posts within excluded areas are exported if there is a valid height in the original model, otherwise a no-data-value is written.

cmL example

```
ens2000/
  ImpExp/
    Type=(Model),
    Export/
      Clear,
      Type=(Grid),
      File=(dtm2000.grd),
      DataFormat=("ArcInfo Ascii Grid"),
      Details,
        RoundTo=("10"),
        OmitBorder=(0);
      OK;
    Close;
```

TIN export

To export the model in TIN data structure select *TIN* from the selection *Type* within the main export dialog window .

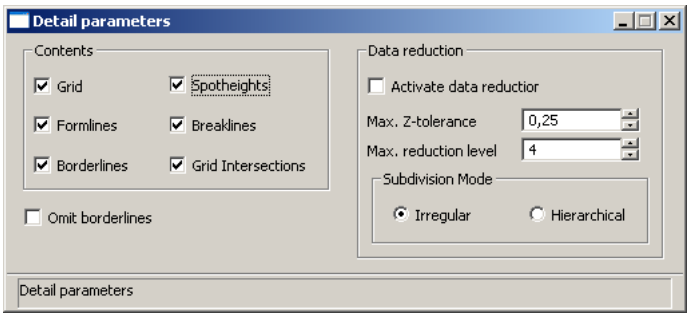


Figure 12.12.: The window *Detail parameters* for TIN export

The following data formats are supported for TIN export:

VRML: Virtual Reality Modelling Language, Format to describe 3D-scenes for the internet, ASCII

STL: Stereo Lithography File Format, ASCII

AutoCAD DXF: Triangle Network as 3DFaces, ASCII

SCOP Winput: Triangles as closed break lines, ASCII or Binary

REB File Format

UCD: Unstructered Cell Data File

2DM: Surface Modelling System 2DM File

CityGrid XML

Within the window *Detail parameters* the selection *Contents* allows to control which features should appear in the exported TIN. The TIN is generated using a Constrained Delaunay Triangulation. Hereby the *Grid* points and *Spotheights* are treated as mass points for the triangulation and the *Formlines*, *Breaklines* and *Borderlines* are considered as constraints. If the checkbox *Grid intersections* is activated line intersections with the DTM grid are exported additionally to the original line vertices, but only if they are not on the straight line connecting the neigbouring vertices. The checkbox *Omit borderline* controls the behavior in excluded areas. If activated all grid posts within excluded areas are included in the TIN, otherwise they are omitted.

One of the most important advantages of the TIN structure in comparison to the raster is the possibility of irregular data spacing. In flat or plain areas it is therefore possible to reduce the number of triangles without (or with little) loss of quality and accuracy of the exported TIN surface. To enable data reduction turn on the checkbox *Activate data reduction*. The basic principle of the applied reduction algorithm is to derive an approximated TIN surface consisting of a subset of the original DTM points. The reduction process can be controlled by two numerical parameters. Use the numeric field *Max. Z-tolerance* to specify the maximum allowed vertical deviation between the original hybrid model and the approximated TIN and the numeric field *Max. reduction level* to specify the number of reduction levels. The reduction process starts with an initial approximation consisting of a coarse regular grid and the structure information of the original hybrid model. The grid spacing Δ_0 of the coarse grid is: $\Delta_0 = g * 2^n$, where g is the grid width of the original model and n is the specified number of reduction levels. The points and lines of the initial approximation are triangulated and subsequently refined by iteratively inserting additional grid points until the maximum allowed z-tolerance is kept. The refinement can either be *hierarchical* or *irregular* by selecting the appropriate item from the selection *Subdivision mode*.

Please note, that there are some restrictions with DTM limits: Arbitrary limit polygons together with data reduction are not yet supported. Thus, a complex user defined limit is automatically reduced to its bounding window if data reduction is activated. Furthermore the user defined grid width as specified in the main export window does not take affect since the data reduction algorithm strictly bases on the original grid structure. The extent of the exported TIN surface is rounded to a multiple of the grid spacing Δ_0 of the coarse grid.

cmL example

```
ens2000/
  ImpExp/
    Type=(Model),
    Export/
      Clear,
      Type=(TIN),
      File=(dtm2000.stl),
```

```
    DataFormat=("STL File Format"),
    Details,
      Spotheights=(1),
      Formlines=(1),
      Breaklines=(1),
      Borderlines=(1),
      OmitBorder=(0),
      ActivateDataRed=(1),
      MaxZTol=(0.5),
      MaxRedLevel=(3),
      SubdivMode=(Hierarchical);
    OK;
  Close;
```

Vector export

To export the model in discrete vector structure (grid points and lines) select *Vector* from the selection *Type* within the main export dialog window .

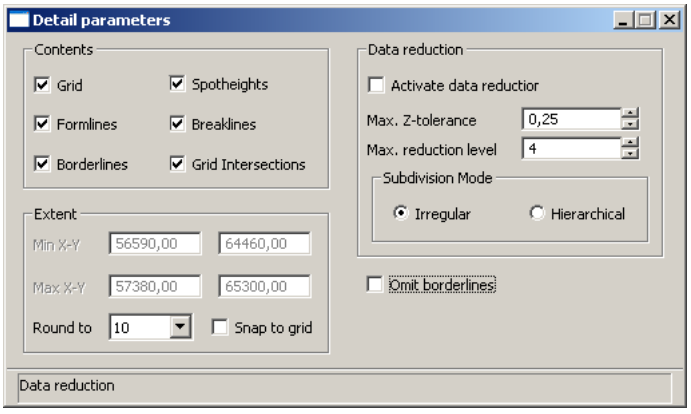


Figure 12.13.: The window *Detail parameters* for Vector export

The following data formats are supported for Vector export:

SCOP Winput: Grid points and line information, ASCII or Binary

XYZ: coordinate-triples, Grid points and line points, ASCII or Binary

AutoCAD DXF: Grid points and spot heights as POINTs, Line information as POLYLINEs, ASCII

VESTRA Winput: like SCOP Winput with special formatting for VESTRA CAD Software, ASCII

Within the window *Detail parameters* the selection *Contents* controls which features of the original model should appear in the exported vector output. Independent activation or deactivation of *Grid*, *Spotheights*, *Formlines*, *Breaklines* and *Borderlines* is offered. If the checkbox *Grid intersections* is activated line intersections with the DTM grid are exported

additionally to the original line vertices, but only if they are not on the straight line connecting the neigbouring vertices.

The group *Extents* allows fine adjustment of the rectangular model area. Activate the checkbox *Snap to grid* to ensure the coincidence of the original and the exported grid if an arbitrary user defined limit polygon was specified (precondition: original grid width = grid width of export file). Alternatively the model extents may be rounded to powers of ten (selection *Round to:*). The checkbox *Omit borderline* controls the behavior in excluded areas. If activated all grid posts within excluded areas are exported if there is a valid height in the original model, otherwise they are omitted.

The data reduction feature as described in the section above is also provided for model export in vector data structure. In this case a reduced set of grid points and thinned structure lines are exported instead of the TIN. For a detailed description of the data reduction algorithm and its parameters please refer to section 12.2.6.

Please note, that arbitrary limit polygons together with data reduction are not yet supported. Thus, a complex user defined limit is automatically reduced to its bounding window if data reduction is activated. Furthermore the user defined grid width as specified in the main export window does not take affect since the data reduction algorithm strictly bases on the original grid structure.

cmL example

```
ens2000/
  ImpExp/
    Type=(Model),
    Export/
      Clear,
      Type=(Vector),
      File=(dtm2000.dxf),
      DataFormat=("AutoCAD DXF"),
      Details,
        Spotheights=(1),
        Formlines=(1),
        Breaklines=(1),
        Borderlines=(1),
        GridInter=(1),
        SnapGrid=(1),
        OmitBorder=(0),
        ActivateDataRed=(1),
        MaxZTol=(0.5),
        MaxRedLevel=(3),
        SubdivMode=(Irregular);
      OK;
    Close;
```

12.2.7. Export Views

Within SCOP++ it is possible to derive different views from digital models. Among them Isolines (section 12.6), Hill Shadings (section 12.7) and Z-Codings (section 12.8)are the

most important. This feature is useful to transfer the results of SCOP++ calculation to other systems. Fore that reason commonly used file interchange formats are supported.

To open the Export dialog window choose *Views* from the selection *Type* within the Import/Export window and press the button *Export...*.

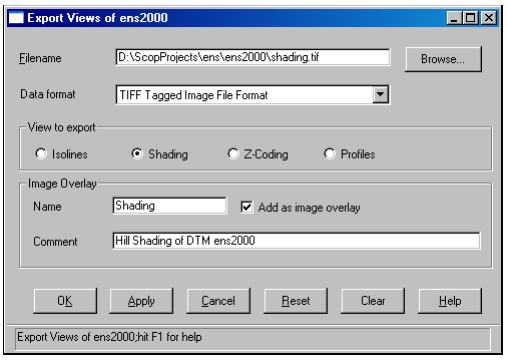


Figure 12.14.: The window *Export Views of overlay*

This dialog comes in two different appearances depending on the availability of modules (see section 1.1). If the package "SCOP++ Analyzer" is not available this window will look like the one depicted, else the window will look like the the figure 12.15.

The dialog window (Export Views of overlay) contains the text field (File) to enter the name of a disk file, the selection (Data Format) to select the appropriate data format from a list of available data formats and the selection *View* to select the type of view to be exported. The filename can also be selected using the file browser . For Hill Shadings and Z-Codings optionally the exported view can be inserted in the current project as an image overlay. For this purpose the group *Image Overlay* contains the checkbox *Add as Image Overlay*. Clicking this checkbox the text field *Overlay* and the text field *Comment* get accessible and the name of an image overlay and an optional comment can be entered there. Clicking the button *OK* or the button *Apply* will start the export process. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones and pressing the button *Clear* will clear all data fields. The button *OK* and the button *Cancel* will additionally close the window . In the case the checkbox *Add as Image Overlay* is activated and the text field *Overlay* is not empty, a new image overlay will be added to the current project. The exported view file will be imported to this overlay automatically.

For exporting Isolines the following data formats are supported:

HPGL (Hewlett Packard Graphics Language): Plotter language of HP, ASCII

SCOP ZWIFI: Proprietary intermediate file format of SCOP, ASCII

AutoCAD DXF: Drawing Interchange File Format of Autodesk, ASCII

HL-Band: Isoline file format of program system TOPSY, ASCII

Data Formats available for image type views (Hill Shading and Z-Coding) are:

GeoTIFF: Both tiled and untiled TIFF is supported. TIFF files written by SCOP++ are always geo-coded.

SCOP PIX: The proprietary image data format of SCOP is tiled and geo-coded.

JPEG: These compressed image data files are not geo-coded.

SCOP pyramid description files: ASCII description files referring to a series of image data files in different resolutions in one of the first three formats.

By selecting the radio button *Profiles* in the selection *Views* (see figure 12.15) the planimetric data of alignments and cross sections (see section 12.10) can be exported. Following data formats are supported:

AutoCAD DXF: Drawing Interchange File Format of Autodesk, ASCII

SCOP WINPUT: Proprietary intermediate file format of SCOP, ASCII. The profiles and cross sections - if available - are stored as form lines.

XYZ: XYZ coordinates of points

Station-point xyz coordinte: Station and XYZ coordinates of points

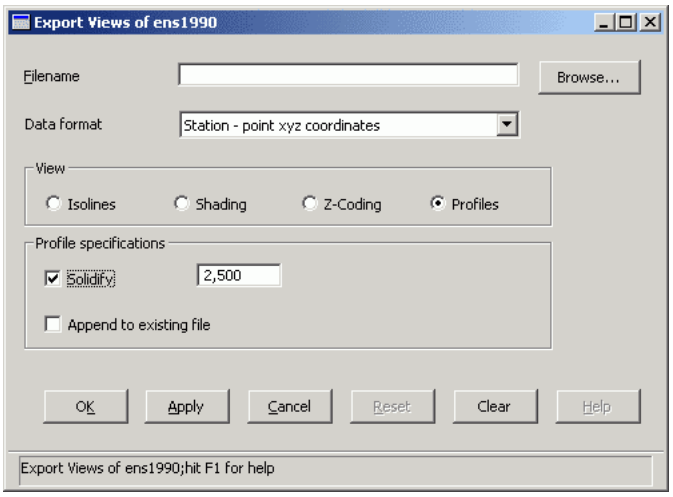


Figure 12.15.: The window *Export Alignment of overlay*

Checking the checkbox *Solidify* will unlock the corresponding numeric field in which the solidifying distance can be entered. The solidifying interval will be constant over the whole alignment (not interfered by the alignment points). If the checkbox *Append to existing file* is checked, the data will be appended to the file specified in the text field *Filename*, else this file will be overwritten without further warnings. The specific formats are described in section 12.10.6.

Please note that an appropriate file extension will be appended if the file name has been entered without extension (ie. .jpg for JPEG files, etc.). The file extension will be adapted automatically in response to a change of the data format. This automatism is applied as

long as a file extension is specified by the user. Note also that specifying a file type in the the file browser does not influence the setting of the data format.

cmL example

```
ens2000/
  ImpExp/
    Type=(Views),
    Export/
      File=(shading.tif),
      Shading,
      DataFormat=("TIFF Tagged Image File Format"),
      AddAsImageOverlay=(on),
      Name=(Shading),
      Comment=(Hill Shading of DTM ens2000),
      OK;
    Close;
```

Note that if the filename is entered without path, then the path to the overlay directory is added automatically. Export of Views is not available for data-only overlays.

12.2.8. Export Secondary Models

This dialog serves for exporting secondary models of the overlay. The extent of the treatment for secondary models differ by the different modules, thus your window *Export Secondary Models of overlay* might look slightly different.

Additionally to the functions outlined in the section Export Model (see 12.2.6), in this window there is a selection which enables the user to determine which secondary model to export.

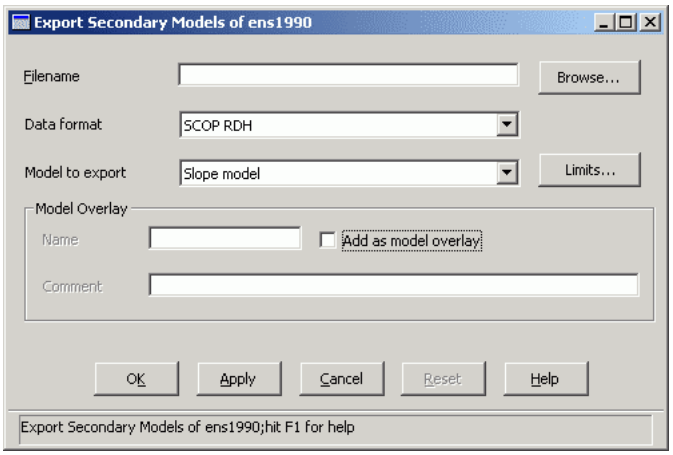


Figure 12.16.: The window *Export Secondary Models of overlay*

In the current version there are two model types to export:

- *Slope model* (see 12.11.1)
- *Nearest model* (see 12.12.1)

12.2.9. Import Secondary Models

In addition to heights of the grid points and the line information (see 17.2 and 5.2.2) a Scop++ DTM file is capable of containing some supplementary information per grid point. Examples for this are:

- Accuracy or quality of the grid point
- Other information concerning one grid point (soil quality...)
- ...

In order to keep the DTM file small these information can be stored only in a twelve step resolution. This resolution will be sufficient for the most cases and values which have a larger domain can be mapped to this resolution by applying a (upto now) linear quantification. In this way arbitrary DTM files can be inserted into existing ones (no matter if this makes sense).

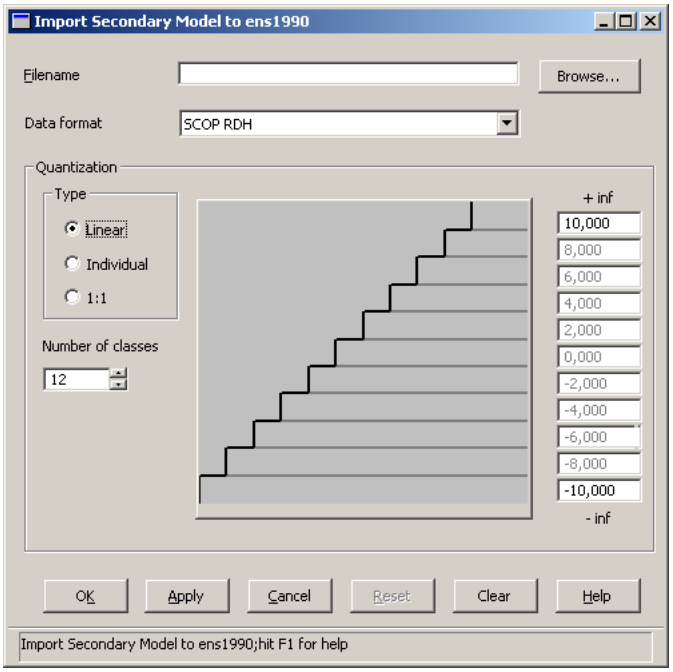


Figure 12.17.: The window *Import Secondary Models of overlay*

In the text field *Filename* the file to import can be entered. Alternatively the file can be browsed via adjoined the file browser *Browse....* In the current version the only format accepted is *SCOP RDH*. The domain of the Z-values of the model to import can be quantified in the group *Quantization*. The type of the quantization is determined by the selection *Type*:

- *Linear*: The domain of the Z-values is divided linear into 12 classes. An upper and lower bound outside which the values will be mapped to the uppermost and lowermost class can be specified in the numeric fields besides the graphics window . The DTM file to import may have the same structure and extent as the file in which it shall be imported.

- *Individual*: All 12 class boundaries can be specified individually. The DTM file to import may have the same structure and extent as the file in which it shall be imported.
- *1:1*: The values from the file which shall be imported will be adopted without changes. The DTM file to import must have the same structure and extend as the file in which it shall be imported.

All changes in the numeric fields will be reflected in the graphics window which visualizes the specified quantization.

The import will be triggered by pressing the button *Apply* or the button *OK*.

12.2.10. Import Parameters

When working on a model overlay the specific parameters for the model interpolation, the isolines, etc. are specified. A complete parameter set can be written to a setup file (export) or read from a setup file (import). The current parameter set is automatically stored on the file *modelOverlay.stp* in the model overlay directory. When a project is opened all available model overlays will be initialized with the parameter settings according to this file. Anyway, the dialog window *Import Parameters* is provided to overtake a parameter set stored on a specific setup file.

To open the Import dialog window choose *Parameters* from the selection *Type* within the Import/Export window and press the button *Import...*.

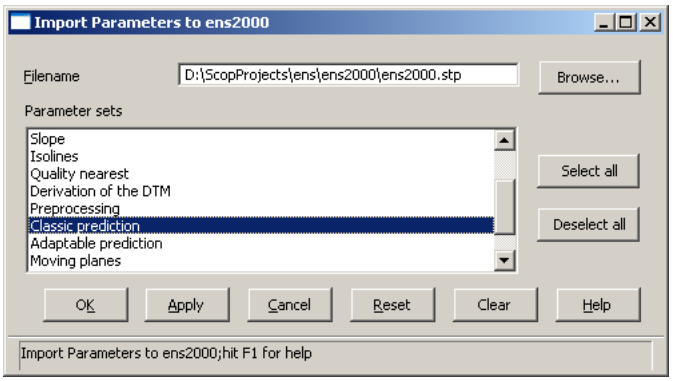


Figure 12.18.: The window *Import Parameters to overlay*

To import a parameter sets enter the name of the setup file into the text field *File*, select one or more types of parameter sets in the selection *Parameter sets* and press the button *OK*. The following parameter sets can be imported from the file: *Data*, *Structure*, *Isolines*, *Shading*, *Z-Coding*, *Profiles*, *Slope*, *Slope vectors*, *Preprocessing*, *Classic prediction*, *Adaptable prediction*, *Moving planes* and *Robust filtering*. The file name can optionally be chosen using the file browser . Button *Select all* serves for selecting all items from the selection *Parameter sets* and button *Deselect all* serves for deselecting all items from this selection.

cmL example

```
ens2000/
  ImpExp/
    Type=(Parameters),
    Import/
      File=(parameters_m100.stp),
      ParameterSets=("Classic prediction"),
      OK;
    Close;
```

12.2.11. Export Parameters

To store the current parameter set to a setup file choose *Parameters* from the selection *Type* within the Import/Export window and press the button *Export...*.

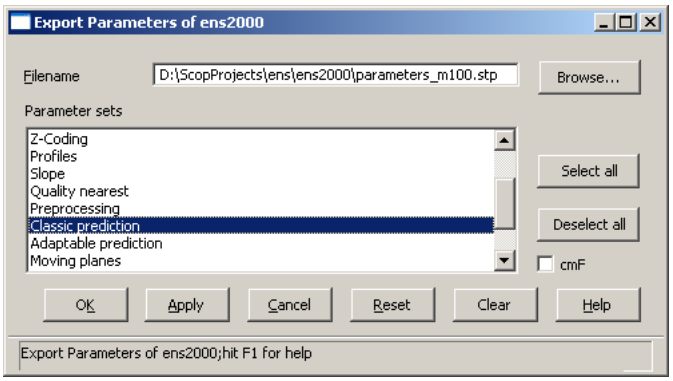


Figure 12.19.: The window *Export Parameters of overlay*

To export a parameter sets enter the name of the setup file into the text field *File*, select one or more types of parameter sets in the selection *Parameter sets* and press the button *OK*. The following parameter sets can be exported to the file: *Data*, *Structure*, *Isolines*, *Shading*, *Z-Coding*, *Profiles*, *Slope*, *Slope vectors*, *Preprocessing*, *Classic prediction*, *Adaptable prediction*, *Moving planes* and *Robust filtering*. Activating checkbox *cmF* allows to written parameters in procedures of the command language otherwise parameters have been written in form of a *SCOP++* setup file. The file name can optionally be chosen using the file browser . Please note that the appropriate file extension will be appended if the file name has been entered without extension. File extension (*.cmF) is append if the checkbox *cmF* is active or file extension (*.stp) is append if the checkbox *cmF* is not active. Button *Select all* serves for selecting all items from the selection *Parameter sets* and button *Deselect all* serves for deselecting all items from this selection.

There are following export/import procedures of the command language *(cmL/cmF)* which can be generate using module export parameters:

@GRD;	Structure
@ISO;	Isolines
@SHD;	Shading
@ZCO;	Z-Coding
@PRO;	Profiles
@SLP;	Slope
@VEC;	Slope vectors
@DTMPreproc;	Preprocessing
@DTMScop;	Classic prediction
@DTMPAK;	Adaptable prediction
@DTMMvP;	Moving planes
@DTMTri;	Triangulation
@DTMROBF;	Robust filtering

cmL example

```
ens2000/
  ImpExp/
    Type=(Parameters),
    Export/
      File=(parameters_m100.stp),
      ParameterSets=("Classic prediction"),
      OK;
    Close;
```

12.3. Model

Button *Model...* is representing the *Digital (Terrain or other) Model*. *Opening* this button (*Properties...*) allows for *Preprocessing* the input data, choosing among the *Interpolation methods*, and manipulating the corresponding parameters. In special cases, preprocessing or interpolation of the digital surface may be done immediately.

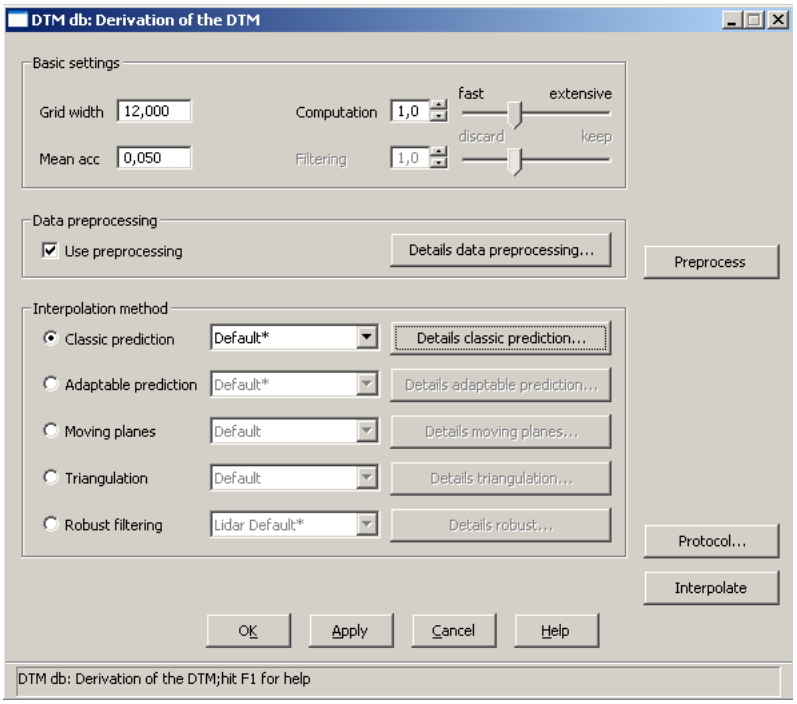


Figure 12.20.: The *Properties...* window *Derivation of the DTM* to button *Model*

The upper level of this hierarchy of *Properties*, the window *Derivation of the DTM*(see fig. 12.20) is to handle just the *most important* specifications concerning DTM interpolation. Special details are to be found one level deeper, under the buttons *Details ....*

12.3.1. Introduction: DTM Structure

Scop applies a hybrid representation of the surface, consisting of a grid representing continuous areas, intermeshed with vector-type data such as break lines or form lines to facilitate the representation of sudden changes in continuity (break lines, highs and lows), or more gradual changes in it (form lines).

Border-, break- and form lines are stored with their original points and their intersection points with grid lines. Spot heights, off-terrain points and control points are stored with their original coordinates.

In large areas without reference points, the CUs are not stored; CUs at the margins are stored in their entirety (even if partially extrapolated due to data lacking outside of the margins). One has to use borderlines for better margins, this way avoiding the stair-case effect due to free-ending computing units.

Rotation of the DTM against the reference system is not possible; however a limiting polygon is useful if the area of interest (e.g. a map sheet) has to be rotated: the minimum-maximum values of the polygon points are then used as model limits (only the resulting rectangle is going to be interpolated).

12.3.2. Basic Settings

The most important parameters for the DTM interpolation. Having selected a *dependent parameter set*, parameters under the buttons *Details ...* will be adapted to them.

Grid Width

The grid width defines the resolution of the digital model. It is the only basic parameter which is applied to all interpolation methods. For attractive views of the model (by views we mean isolines (being contour lines for elevation models), hill shading and the similar), grid width should be about 1.5 mm in the image thus produced – i.e. in the map, or on the screen.

Default for the grid width is derived by analyzing data density. In the current version, the values thus derived are only proper for photogrammetric data acquisition for purposes of topographic mapping. Therefore, at least in all other cases, *it is of utmost importance to check on the value of the grid width, and to correct it so to fit the actual application*.

Mean Accuracy

This value is used as the filter value of bulk data. The filter values of spot heights, form and break lines are adapted to that at ratios of 5:1:3:2. In *Hierarchic robust filtering*, additionally, the weight function of filter steps and the upper and lower elimination distances of sort out steps will be varied.

Computation

This slide lets users control the compromise between a faster and a more extensive computation. More exactly, the extents of computing units are varied. In *Hierarchic robust filtering*, the sizes of computing units of filter and interpolation steps are adapted.

Filtering

This parameter is only accessible having selected *Hierarchic robust filtering*. The weight functions of filter steps and the filter values of bulk data of filter and interpolate steps are varied. This results in rather discarding or keeping questionable points.

12.3.3. Data Preprocessing

This option may be used to process the input data before the interpolation starts. The results will only change the input temporarily for the time of interpolation. Having selected *Hierarchic robust filtering*, it is not accessible. See section 12.3.6 for a detailed description.

12.3.4. Method of Interpolation

In this version, there are - dependent on the available modules (see 1.1) - up to five methods of interpolation provided. Each method holds a drop-down list to select a parameter set for interpolation. This set is kept identical to the one under the buttons *Details ...* . A *dependent parameter set* is indicated by an '*' (asterisk) at the end of its name.

Classic Prediction

This is the default interpolation method this version of *SCOP++*. It is applying *SCOP.DTM* of `SCOP V3.5` as a background server to perform the computation. For this method, a rich set of parameters is provided to tune the interpolation according to the specifics of the surface, of data distribution, and of the user's special needs (see section 12.3.7).

Adaptable Prediction

This is an improved organization of the linear prediction. It applies variably sized computing units for interpolation, depending on a thorough analysis of local data distribution. The resulting model will be written onto the disk to carry computing units of constant size, so to correspond to the classical *Scop Dtm* structure.

Moving Planes

This method of interpolation is included in the package "SCOP LIDAR". This method is by far less time consuming than linear prediction but produces a surface less smooth. Therefore it is the method of choice for eliminating data errors and for testing. A detailed explanation of the theory behind this method can be found in section 19, the user interface is described in section 12.3.9.

Triangulation

In this method, the input data is interpolated through a Delaunay triangulation, where break and form lines are inserted as constraints. Thus, this method produces a waveless, but jagged surface. It is the method of choice for exclusively manually (therefore selectively) collected data, e.g. by tachymetry.The user interface of this method is described in section 12.3.10.

Hierarchic Robust Filtering

This method of interpolation is included in the package "SCOP LIDAR". It is recommended, if data with gross errors shall be used to create a terrain model free of these gross errors. It is also applicable to generate a terrain model from a reduced data set. This method is most often applied to data from airborne laser scanners. A detail explanation of the theory behind this method can be found in section 20, the user interface is described in section 12.4.

Immediate Preprocessing / Interpolation

According to the principle of *lazy processing* (see section 5.3.1) any DTM will only be interpolated, if it is needed for some other process – e.g. for displaying isolines. The same holds for preprocessing, as preprocessing is done only when a DTM has been requested. Should the user prefer to let the data be preprocessed, or the model be interpolated immediately, she should click at button *Preprocess*, or button *Interpolate*, respectively.

Protocol

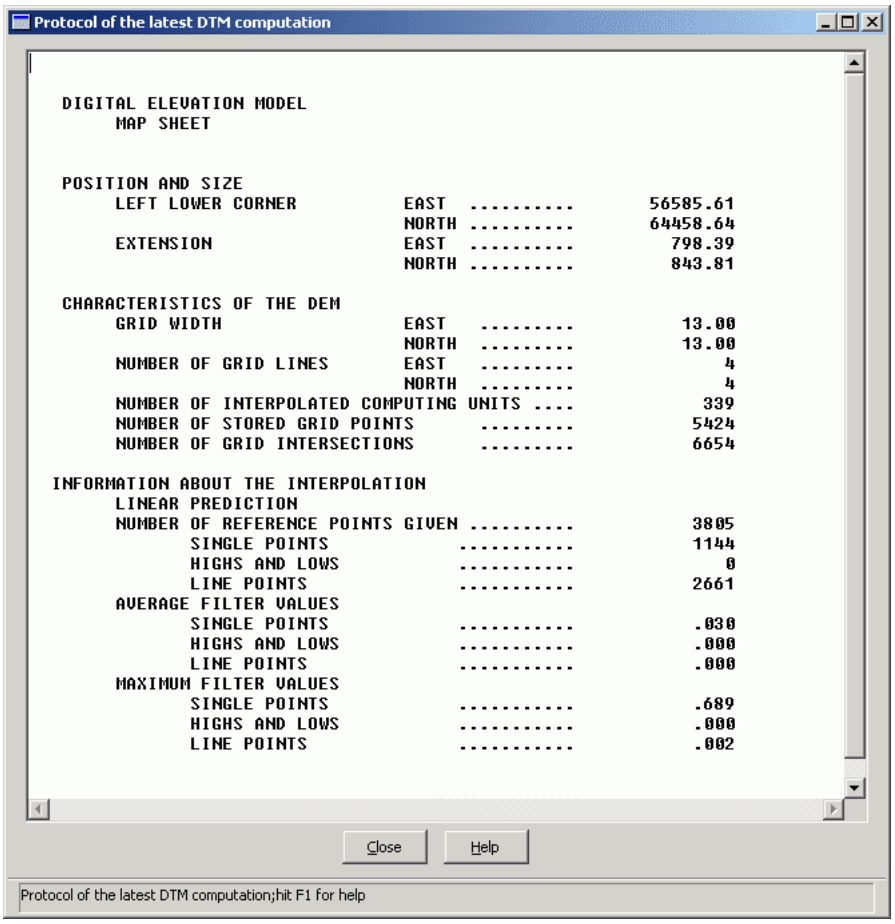
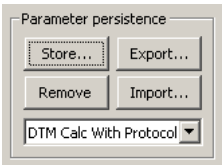


Figure 12.21.: The window *Protocol view*

Below the state-switch *Protcol...* you will find an excerpt of the latest interpolation. In the depicted figure 12.21 you see for example the output of the `SCOP V3.5` server.

12.3.5. Parameter Persistence



The parameter windows for the different interpolation methods provide a possiblity to store and retrieve parameters proven to suit a particular kind of terrain, data and similar. Via the group *Parameter persistence* situated in the upper right of the windows, a name for a parameter set can be specified which will be the key for storage as well as for retrieval. This name is to be entered in the appearing dialog beneath the button *Store* in this group. On confirming the name by pressing the button *Ok*, this name will be appended to the list of the available parameter sets, which can be inspected in the selection below the button *Remove*. Via this selection a stored parameter set can be retrieved, by selecting the name under which it was stored. In the reverse direction, the parameter set whose name is active in the selection , will be discarded from the list of parameter sets to store, on pressing the button *Remove*. It is also possible to write the current parameter set to a setup file (export) or read it from a setup file (import). To open the Export dialog window press the button *Export* and to open the Import dialog window press the button *Import*. Those dialogs look similar to the *Import/Export Parameters* dialogs which are described in sections 12.2.10 and 12.2.11.

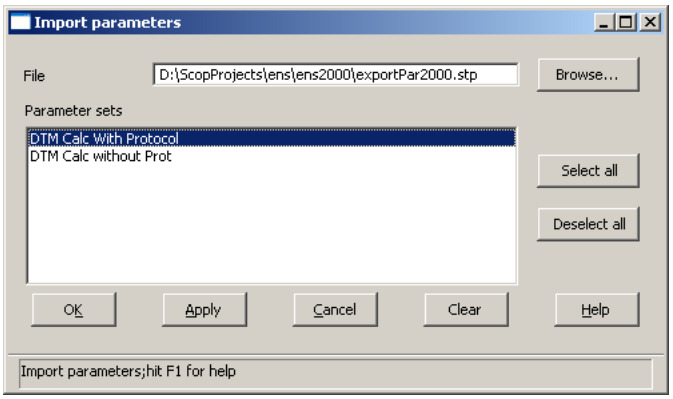


Figure 12.22.: The window *Parameter Persistence - Import Parameters*

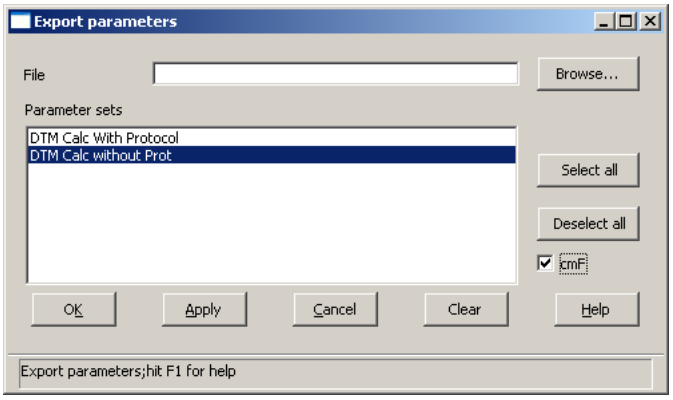


Figure 12.23.: The window *Parameter Persistence - Export Parameters*

To import or export the parameter sets enter the name of the setup file into the text field *File*, select one or more types of parameter sets in the selection *Parameter sets* and press the button *OK*. List of the parameter sets in the selection *Parameter sets* in case of export is the same as the list in the selection from the *Parameter Persistence* group. Otherwise the list of the available parameter sets in case of import is generated from import file. Button *Select all* serves for selecting all items from the selection *Parameter sets* and button *Deselect all* serves for deselecting all items from this selection. Activating checkbox *cmF* in the window *Export Parameters* allows to written parameters in procedures of the command language otherwise parameters have been written in form of a *SCOP++* setup file. The file name can optionally be chosen using the file browser . Please note that the appropriate file extension will be appended if the name of the export file has been entered without extension. File extension (*.cmF) is append if the checkbox *cmF* is active or file extension (*.stp) is append if the checkbox *cmF* is not active.

12.3.6. Detailed Parameters for Data Preprocessing

Clicking at button *Details data preprocessing* opens a window, where detailed parameter settings can be made to adapt the preprocessing of input data according to its characteristics (see Fig. 12.24).

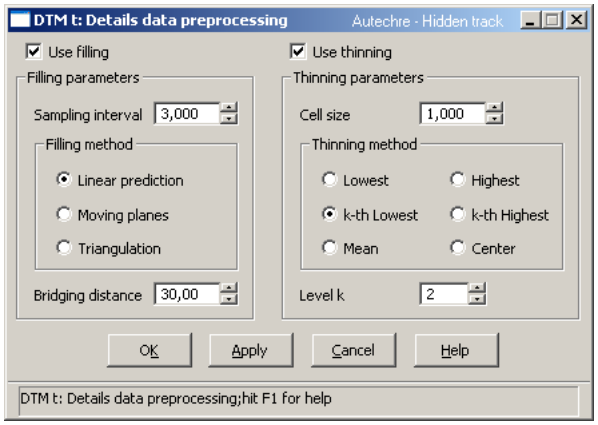


Figure 12.24.: The window *Data preprocessing*

Thinning

When using *thinning*, a regular grid is laid over the whole *Model* area. From each cell of this grid, one point is extracted and then used as input for *Filling* (if activated), or DTM interpolation, respectively.

- *Cell size* specifies the side length of each cell.
- *Thinning method* defines which point to extract from each cell: the *highest*, *lowest*, *k-th Lowest*, *k-th Highest*, *Mean*, or *Center* point. Using method *Center*, the point that resides closest to the center of a cell is selected. The method *Mean* selects the center of mass of each cell, i.e. not an original data point. See section 12.4.3 for further explanations.

- Having selected *k-th Lowest* or *k-th Highest*, *Level k* specifies which point to extract exactly: second lowest/highest, third lowest/highest, and so on.

Break and form lines are not considered in *thinning*, but they are output from *thinning* for forthcoming processing (i.e. either *filling*, or DTM interpolation). The output file of *thinning* is called <modelOverlayName>_preThin.*b*wnp and can be found in the overlay directory. (Remark: This is only an intermediate file. If the model is re-interpolated without preprocessing a possibly existing file from a previous run will be deleted.)

Filling

This method may be applied to fill data voids with a regular grid of filling points, e.g. in already filtered Lidar data sets. Points will be inserted only on areas where the distance to the next original data point exceeds the *Sampling interval*.

- *Sampling interval* specifies the distance between filling points.
- *Filling method* is the way how filling point heights are calculated:
  - *Linear prediction* results in a smooth, undulating run of heights
  - *Moving planes* is the fastest method, but may yield a rough surface
  - *Triangulation* produces a waveless, but jagged run of heights
- In the center of large data gaps, the heights of filling points may be extrapolated too far. *Bridging distance* specifies the threshold for the distance to the next original data point above which no filling points will be inserted.

Break and form lines in the input data are used in all methods. Using *Linear prediction* it may be necessary to simplify the line information, which is done automatically, according to requirements. If interpolation still fails, the user is asked to disregard lines completely in the estimation of filling point heights.

The output file of *Filling* is called <modelOverlayName>_preFill.*b*wnp and can be found in the overlay directory. (Remark: This is only an intermediate file. If the model is re-interpolated without preprocessing a possibly existing file from a previous run will be deleted.)

12.3.7. Detailed Parameters for Classic Prediction

Clicking at button *Details classic prediction* opens a window, where detailed parameter settings can be made to adapt the interpolation method according to the characteristics of the input data (see Fig. 12.25).

'Net' and 'Gross' Computing Units

Before going into further details, we have to clear the meaning of the above terms.

DTM interpolation is a patch-wise process. Patches are termed 'Computing Units´ or 'CUs´. They form a regular pattern of non-overlapping squares (or, non-square grids, of rectangles). For interpolating the DTM surface for a computing unit, points within their

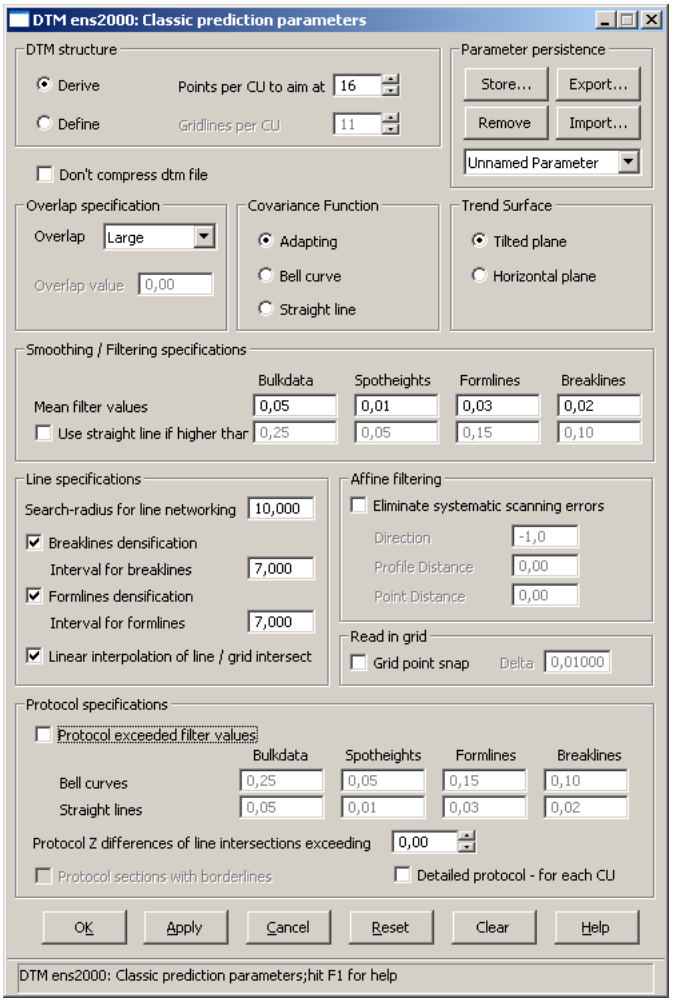
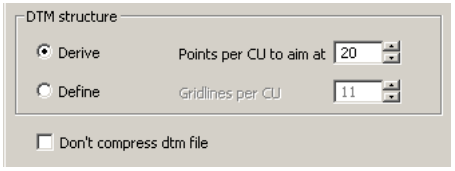


Figure 12.25.: The window *Classic prediction*

area, and also from their surroundings are applied (see below). The area covered by this cloud of points is termed 'Gross computing unit'. Thus, 'Gross CUs' are overlapping areas.

Computing Units are subdivided by break lines crossing them into two or more *Interpolation Areas.*

DTM structure



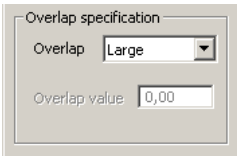
Choosing radio button *Derive* results in automatic determination of CU size based on point density.

The numeric field *Points per CU to aim at* allows for specifying a number of reference (terrain) points per CU to be considered as best in this iterative process. The average number of points in the CU is typically between 4 and 25.

Choose radio button *Define* for to explicitly define CU size. numeric fields *Gridlines per CU* specify the number of grid lines per CU ($3 \leq n \leq 49$);

If in an area of 5*5 CUs there are less than 3 reference points covering the surface of the middle CU, this CU is not computed. This means that small CUs and irregularly distributed reference points may lead to gaps in the *Dtm*. On the other hand, there are some limitations (see section 12.3.12) that might be violated using large CUs.

Overlap specification



The overlapping area of the clouds of reference points applied to interpolate neighboring CUs is usually computed for each CU individually depending on reference point distribution; the attributes SMALL, MEDIUM, LARGE, XL and XXL allow for influencing this process.

The default is OVERLAP=LARGE. Should this result in views (especially in hill shading) indicating computing unit borders, one can try to change this specification – and also, in some cases, the grid width.

For special cases it is possible to choose OVERLAP=CONSTANT and explicitly specifying the width of the overlap using numeric field *Overlap value*.

In the gross CU (CU including overlapping area) there should be at least nine, in the net CU not more than 36 points, so the ratio to be handled by this program without problems is in an extreme case

$$\frac{9 \text{ points}/25 \text{ CUs}}{36 \text{ points}/1 \text{ CU}} = \frac{1}{100}$$

in terms of the mean number of points (and 1/10 in terms of distance) between the areas of lowest and highest point densities. Concerning interpolation quality, however, this ratio should be not less than

$$\frac{9 \text{ points}/9 \text{ CUs}}{36 \text{ points}/1 \text{ CU}} = \frac{1}{36}$$

in terms of the mean number of points (and 1/6 in terms of point distance).

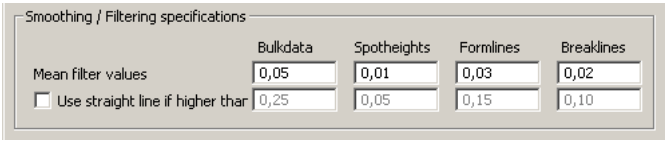
Further points of view when influencing the size and overlap of CUs:

- These specifications will also influence computing time. It increases rapidly with the number of reference points per CU (including overlap).
- The high accuracy expected of Linear Prediction forbids using OVERLAP=SMALL.
- When filtering (and specifically when filtering scanning errors), choosing somewhat larger CU size (e.g. 16 points in the average net CU) and OVERLAP=LARGE may improve results.
- When filtering scanning errors, OVERLAP=XL should be considered.

Do not Compress the *DTM*

In special cases, selecting this checkbox will enhance the resolution of representing elevations. This is hardly needed at all.

Smoothing/Filtering Specifications



This is a table for specifying filter values and different thresholds for the interpolation process.
In the numeric fields besides *Mean filter values* average filter values can be required for different data classes (*a-priori* values). With checkbox *Detailed Protocol File* active (see section 12.3.7), the actual (a-posteriori) values will be listed in the protocol file as written by the `SCOP V3.5` server. Comparing a-priori with a-posteriori values allows for fine-tuning the process by re-specifying these values and repeating the interpolation (an action needed in seldom cases only). The proper values should reflect the accuracy of data acquisition, or otherwise the measure of smoothing as required.

If a straight line as base function (see section 12.3.7) is defined, no filtering takes place except where necessary to avoid contradictions in the interpolated surface.

If affine filtering is applied (see section 12.3.7), the filter values are used only when the scan direction cannot be determined.

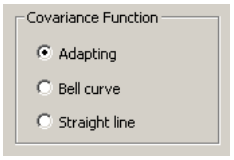
If a considerable amount of smoothing is desired, the parameter a large overlap (see section 12.3.7) should be used to improve the result.

In the numeric fields besides the checkbox *Use straight line* upper limits for actual filter values, when the bell curve is used as base function, can be specified. If these limits are exceeded, the interpolation is repeated using the straight line as base function. They can specified for the same data classes as the average filter values and will only applied if the checkbox *Use straight line* is checked. Default values for these maxima values are derived automatically as the fivefold value of the mean filter values as long as the user hasn't changed them.

If affine filtering is applied (see section 12.3.7), these thresholds are omitted.

With checkbox *Detailed Protocol File* active(see section 12.3.7), the corresponding computing units can be identified in the protocol for interpolation.

Covariance Function



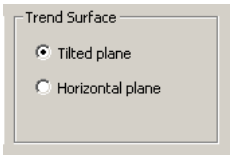
In the group *Covariance Function* it can be specified which base function will be used for the linear combination to get the prediction function. Two base functions are used: (x = distance, m = function parameter)

$$g_c = \frac{1}{1 + \frac{x^2}{m^2}} \qquad \text{(bell curve)}$$

$$g_l = 1 - |\frac{x}{m}| \qquad \text{(straight line)}$$

- Selecting radio button *Bell Curve* requests the bell curve to be used as base function in all CUs. This results in smooth interpolation surfaces with regular curvature, and is well suited for topographic purposes. However, using the bell curve as base function, very irregular terrain features cannot be reproduced properly.
- Selecting radio button *Straight Line* requests the straight line to be used as base function in all CUs. This results in minimal filtering, and in changes of curvature at reference points. This allows in most cases the interpolated surface to go through the reference points even in rough irregular terrain, except when two reference values are nearly identical concerning their horizontal situation. This base function has advantages when looking for errors in data acquisition, and for some special purposes.
- Selecting radio button *Adapting* requests the Straight Line as base function in CUs containing break lines, and the Bell Curve elsewhere; this is the default case. It results in sharp representation of areas with break lines, and in smooth surfaces elsewhere.

Trend Surface



The first step in the calculation process is to remove some trend from the data points within each interpolation area. Two types of trend surface may be selected.

- Choosing radio button *Tilted Plane* requests a tilted plane to be used as trend surface in each interpolation area. This is default.
- Choosing radio button *Horizontal Plane* requests a horizontal plane to be used as trend surface in each interpolation area.

If in some interpolation area the elevations of the Tilted Plane as derived exceed the extreme reference values within the area, the program will automatically switch locally to the Horizontal Plane.

Line specifications

In the group *Line specifications* the following options are available:

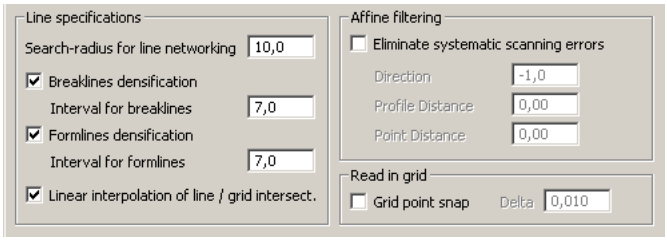


Figure 12.26.: The groups *Line specifications, affine filtering and read in grid*

- The numeric field *Search radius for line networking* allows for specifying a threshold preventing ending points of *break lines* to be connected with adjacent *break lines*. Default is the *grid width*. Usually it is recommended to set this value to 0.01 times grid width to avoid unwanted connections.
- The checkbox *Breakline densification* together with the numeric field *Interval for breaklines* allow to enforce the densification of points along breaklines *before* performing surface interpolation: where the gap between two adjacent points exceeds the value specified for, additional points will be inserted by applying linear interpolation between the neighbors. Default is 0.7 * grid width
- The checkbox *Formline densification* together with the numeric field *Interval for formlines* controls a similar process for *form lines*.
- The checkbox *Linear interpolation of line/grid intersections* prevents filtering along break lines: before performing linear prediction, all intersections of break lines with the grid will be computed applying linear interpolation along the breakline. This representation of the break line will be the final one. All points along it will participate in deriving the interpolating function for the adjacent interpolation areas; these points will get a high weight in this process.

  This allows for a very high emphasis of break lines. This provides also preference for the line before other data situated adjacent to it.

  Without this option, grid intersections are interpolated the same way as the all grid points (i.e. by linear prediction), in most cases resulting in filtering accidental errors in data and smoothing the surface.

Read in grid

The checkbox *Grid point snap* is needed when reading in data acquired (or computed externally) in form of a regular grid pattern. For this:

- the *grid width* has to fit that of the grid to be imported (however, the sequence of the points to be input is arbitrary).
- In the *SCOP++* window *Limits*, a window for Model has to be entered (preferably numeric entries, see section 4.4), so that at least the lower left corner fits exactly a point of the input grid.
- In group *Size of the Net CU* the radio button *Derive* has to be chosen, and the numeric field *Gridlines per CU* should be entered as 2 through 12.

- The *Overlapping of Gross CUs* should be chosen as *small*.
- The numeric field *Delta* can be used to specify a distance from the assumed grid point above which points won't be used.

Affine filtering

Affine Filtering corresponds to performing linear prediction using an elliptically rotated bell curve as covariance function. This ellipse can be produced by an affine transformation of the rotational circle – or, and this is what actually happens, by applying the corresponding affine transformation to the reference points before the filtering (thus moving them closer to each other in one direction) before the interpolation, and moving them back by performing the inverse transformation after it.

The parameters of such affine transformation will be derived based on the three values entered to the numeric fields *Direction, Profile Distance, and Point Distance* – another image of this process. The first value determines a direction (as the angle in grades between the East and the direction, counter-clockwise), the second parameter influences smoothing *across* this direction, and the third parameter determines smoothing along it (think of both axes of an ellipse):

- *Increasing* the value for *profile distance* results in a smoother surface across to the *Direction* (of the "profile");
- increasing the value for the *Point distance* (along the "profile") results in a smoother surface in profile direction.

Affine Filtering is performed for each CU individually. The covariance function in such computing units must be specified as Bell Curve (i.e. if *Adapting* has been chosen, Affine Filtering will not take place in CUs containing break lines).

If Affine Filtering in some computing unit is not successful, standard linear prediction will be performed.

*In a special case, when data patterns correspond to a grid or parallel profiles, and are coded as such, then the program can derive values for the parameters as described (*Direction, Profile Distance, and Point Distance *automatically:*

- If numeric field *Direction* is given a negative value, the program will attempt to derive profile direction from the reference points.
- If the values for numeric fields *Profile Distance and/or Point Distance* are specified as 0, the respective values will be derived from the reference points.

For this to function in a computing unit, it has to contain at least two profile sections with three points in each.

Protocol specifications

In this group there are different means to specify the extent of the protocol written by the `SCOP V3.5` server. Please pay attention that the checkbox *Protocol* (see section 9.2.8) has to be checked to produce an output of protocol files.

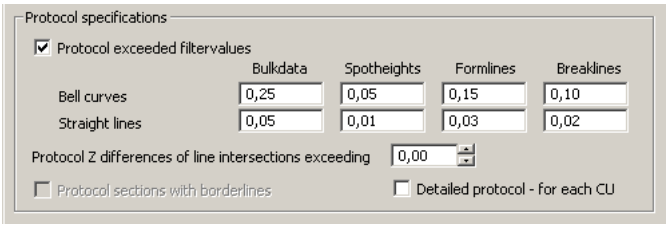


Figure 12.27.: The group *Protocol specifications*

- The checkbox *Protocol exceeded filtervalues* allows turning on and off the protocol for filtervalues.
- The numeric fields besides *Bell curves* state the filter values above which the points (Bulkdata, Spotheights, etc...) will be protocolled if a bell curve is used as covariance function.
- The numeric fields besides *Straight lines* state the filter values above which the points (Bulkdata, Spotheights, etc...) will be protocolled if a straight line is used as covariance function.

Both - the numeric fields besides *Bell curves* and the numeric fields besides *Straight lines* - will be updated automatically with the fivefold value of the mean filter values as long as the user hasn't changed them.

- The numeric field *Protocol Z differences of line intersections exceeding* defines the threshold above which line intersections with Z differences exceeding the value specified will be listed on the protocol file. The checkbox *Protocol sections with borderlines* allows to turn on/off the protocol for those.
- The checkbox *Detailed protocol - for each CU*. If this is checked, for each CU there is printed one line containing the detailed information related to CU plus overlap onto file `<overlayName>.out`. This file is written by the `SCOP V3.5` server. In Chapter *Getting Started* 6.3 it is proposed to place a symbol of some editor onto the GUI to facilitate inspecting this file. Its contents are as follows.

```
1. Read in directive
2. Name of the project
3. Read in or computed values for grid width, CU size, Dtm size
   3.1 For each iteration step to compute CU size
         CU size, n.o. grid lines, max.n.o. ref. points/CU,
         max.n.o. lines,
         n.o. CUs not near the borders,
         mean n.o. ref. points in these CUs,
         max.n.o. ref.points/CU including line densification.
   3.2 Density model of reference points
         Matrix of the CUs, one character for each CU
           space: CU contains no points
           0    : CU contains 1-9 ref. points
           1    : CU contains 10-19 ref.points
           etc.
           *    : CU contains more than 99 ref. points
   3.3 Density model of the lines
```

```
          Similar to 3.2, except that digit = n.o. lines in CU,
            * : contains more than 9 lines
            Lines intersecting the CU without a point in the CU
            are not counted.
4. N.o. ref. points and lines
5. Line intersections, if any (and parameter SECTIONS given):
     Intersection point (east, north); for each of the two lines:
       Structure number, number of point, z coordinate at
       the intersection (interpolated linearly);
     difference of the two z coordinates.
6. Lowest and highest z coordinate of the ref. points
   used in the Dtm.
   6.1 Matrix of the CUs, one character for each CU:
         .  : CU contains no ref. points and will not be computed
         X  : CU contains ref. points and will not be computed
         0  : CU contains no ref. points but will be computed.
              because there are enough ref. points in overlapping
              areas;
         +  : CU contains ref. points and will be computed.
7. If contradictions between borderlines are found, the
   coordinates (reduced to the Dtm left lower point) of the
   point,where the contradiction was found is printed.
8. Table of interpolation details, per CU:
   ------------------------------------------------------------------
   NE     CU index (column)
   NN     CU index (row)
   NS     Number of reference points after densification of lines and
          subdivision of interpolation areas (points on break lines
          count twice)
   NL     (Number of break and form lines) + 1
   NR     Number of border line points
   BUL    Number of bulk points
   H+L    Number of spot heights
   STR    Number of form line points
   BRE    Number of break line points
   ZMIN   Lowest reference value
   ZMAX   Highest reference value
   NT     Number of interpolation areas
   PRED   Interpolation
          C  base function is bell curve
          L  base function is straight line
          T  trend surface used (very little differences between trend
             surface and reference values)
          S  elimination of scanning error
   GRMIN  Lowest grid or grid intersection value
   GRMAX  Highest grid or grid intersection value
   ------------------------------------------------------------------
```

OK and Interpolate

Clicking at button *OK* will make the current specifications valid. An interpolation will, however, only occur, when the model to be derived is immediately needed for some other

process – e.g. for displaying isolines (for this, the state of the corresponding state-switch should be set to *Display*).

A commandLine / commandFile (cmL/cmF) example for this window is:

cmL example

```
ens2000,
  model,
    Gridwidth 6,
    ClassicPrediction,
    DetailsClassicPred/
    // selecting classic prediction via SCOP.DTM Servers
    Filter,
    // Select filter specifications
    UseStraightLine 1,
    FilterBulkdata = 0.02 0.02 0.04 0.05,
    // Access of the mean filter values via specification
    // chaining
    FilterBulkdataMax = 0.02 0.02 0.04 0.05,
    // Access of the threshold values via specification
    // chaining
    OK;
  OK;
shade display, z-code display;
```

12.3.8. Detailed Parameters for Adaptable Prediction

Clicking at button *Details adaptable prediction* opens a window, where detailed parameter settings can be made to adapt the interpolation method according to the characteristics of the input data (see Fig. 12.28).

Overlap Specification

The three items

- One gridrow
- Two gridrows
- No overlap

in this selection can be used to specify the overlap of the 'Gross' Computing Units (see section 12.3.7). With the numeric field *N'th point* the number of points which are used can be determined. A value of one means that every point in a computing unit which is in the gridrow specified in the selection *Overlap specification* is used. If the item *Two gridrows* is selected, every point from the first and every n'th point from the second gridrow is used.

If checkbox *Reduce overlap* is selected then the number of points within the overlapping area will be reduced to a reasonable amount. This may reduce the computing time for areas with bad point distribution.

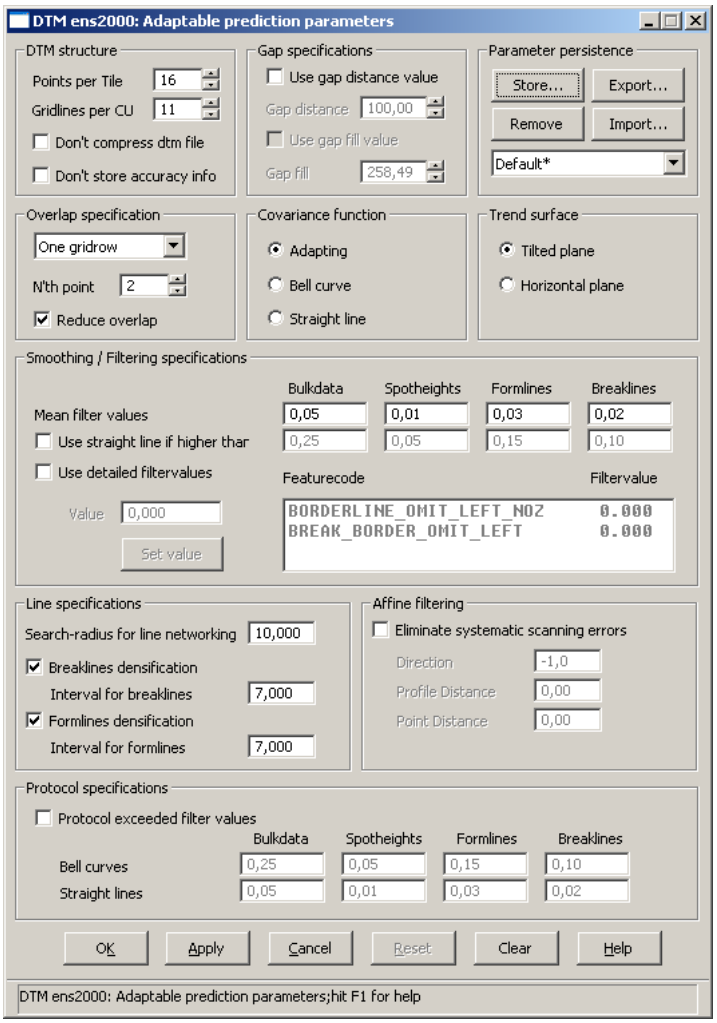


Figure 12.28.: The window *Adaptable prediction*

Smoothing/Filtering

In numeric fields besides *Mean filter values* average filter values can be required for different data classes (*a-priori* values).

Individual filter values for data classes which exceed this classical scheme, can be determined via the widgets below these numeric fields . To enable the so-called detailed filter values the checkbox *Use detailed filtervalues* has to be checked which makes the GUI elements needed for this task accessible. The selection *Featurecode Filtervalue* is a list of all defined featurecodes (see section 9.4.2)for which data is available (in the current overlay). The filtervalues for the featurecode can be specified be selecting the items for which the filtervalues should be specified, then typing in a filtervalue in the numeric field *Value* and pressing the button *Set value*. The filtervalues for the corresponding featurecodes on the left side of the selection will be updated and used in a following adjustment.

RDH Structure

This group determines the persistent characteristics of the DTM-file(see section 17). The numeric field *Gridlines per CU* determines the number of gridpoints stored. Via selecting the checkbox *Don't compress the DTM* the DTM will be stored in an unpacked Version, this needs about twice the disk storage of the packed Version. In some cases, the unpacked DTM may be necessary for accuracy reasons. Via selecting the checkbox *Don't store accuracy info* no accuracy info will be stored in the DTM.

Gap Specifications

The numeric field *Gap distance* determines the maximum distance to nearest reference point. If this distance exceeds the given value, the the gridpoint is set void. It is only accessible if the checkbox *Use gap distance value* is selected. When selecting the checkbox *Use gap fill value* a height may be specified with numeric field *Gap fill* which will be used for void gridpoints. If this parameter is not specified void gridpoints will get an interpolated or extrapolated height whenever possible; if this not possible (e.g. outside the x/y-min/max area) the gridheight is set to the minimum height of the DTM.

For the other parameters see details in section 12.3.7.

12.3.9. Detailed Parameters for Moving Planes

Clicking at button *Details moving planes* opens a window, where detailed parameter settings can be made to adapt the interpolation method according to the characteristics of the input data (see Fig. 12.29).

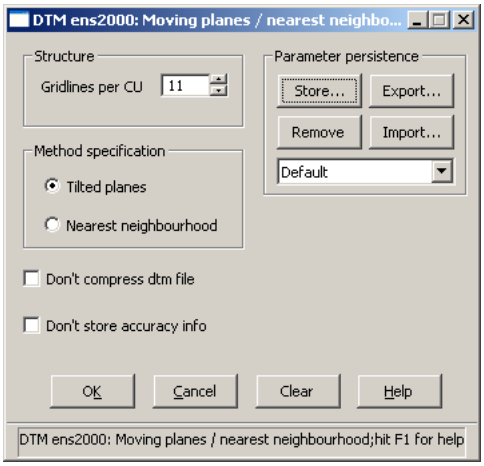


Figure 12.29.: The window *Moving planes*

Structure

This group determines the persistent characteristics of the DTM-file. The numeric field *Gridlines per CU* determines the number of gridpoints stored (see section 17).

Method Specification

This group provides two kinds of interpolation:

- Tilted planes
- Nearest neighbourhood

Via selecting the checkbox *Don't compress the DTM* the DTM will not be compressed, thus resulting in a much larger file. Via selecting the checkbox *Don't store accuracy info* no accuracy info will be stored in the DTM.

12.3.10. Detailed Parameters for Triangulation

Clicking at button *Details triangulation* opens a window, where detailed parameter settings can be made to adapt the interpolation method according to the characteristics of the input data (see Fig. 12.30).

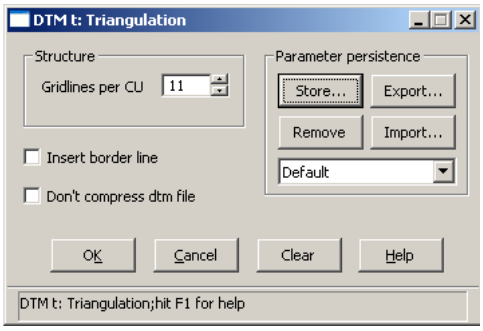


Figure 12.30.: The window *Triangulation*

Structure

This group determines the persistent characteristics of the DTM-file. The numeric field *Gridlines per CU* determines the number of gridpoints stored (see section 17).

Having selected the checkbox *Insert border line*, the convex hull of the triangulation is inserted into the DTM as border line. Via selecting the checkbox *Don't compress the DTM* the DTM will not be compressed, thus resulting in a much larger file.

Memory Considerations

The triangulation requires altogether about 88B of memory for each inserted point. Assuming e.g. a computer with 2GB RAM, triangulating 20 million points therefore should be no problem.

12.3.11. Object Shapes

Line data with encoding 'ObjectShape' (cf. e.g. Table 12.1) are processed in a special manner during model interpolation. In fact, ObjectShapes are solely employed at the end of interpolation, to the otherwise finished surface. Along closed ObjectShapes, jump edges are inserted into the surface. While the model outside the ObjectShape remains unchanged, all grid point heights inside are set to the height of the Delaunay triangulation of the ObjectShape. Moreover, all lines in the ObjectShape's interior are discarded. This way, ObjectShapes facilitate the representation of closed, vertical structures in a surface model.

12.3.12. DTM Restrictions

There are a few restrictions concerning the calculations of a DTM with SCOP++. They can be separated into groups:

- DTM-wise and
- CU-wise restrictions.

They are listed in the following table:

Restriction		Error Message(s) of the SCOP Server
DTM		
Max. N.o. breaklines	131071	Errormessg. 2037
Max. N.o. formlines	131071	" 2036
Max. N.o. borderlines	131071	2038
Max. N.o. points per line	4095	2035
Max. N.o. control points	100	only max. n.plotted
Max. N.o. CUs (including CUs that are not computed)	1*10**9	Errormessg. 2042
CU		
Max. N.o. gridlines(east, north)	49	Errormessg. 2011 2012
Min. N.o. gridlines	3	" 2005 2006
Max. N.o. ref.points	1600	" 2053
Max. N.o. ref.points including overlap	1800	" 2064
Max. N.o. ref.points for PREDICT, incl.densificatio	2500	" 3106 3108
Max. N.o. ref.points for FAST, incl.densification	1600	" 3121
Max. N.o. form- and breaklines incl. overlap	150	" 2065
Max. N.o. borderline points incl. overlap	400	" 2066
Max. N.o. breaklines (PREDICT)	123	" 3107
Max. N.o. interpolation areas (PREDICT)	255	" 3105
Max. N.o. steps in gridreduction	3	" 2020
Max. words for one CU = N.o.grid points * 3 + (n.o.line points + n.o.grid intersections + n.o.lines + n.o.special points)* 4	7900	" 3104 3109

12.3.13. Error Messages

An error which occured during DTM calculation will be reported via an error message which has the hint to check the corresponding log file (see section 9.2.8). Provided that the protocol option is enabled this will contain a diagnosis of the error. The following error-number, text/comment table is a thorough description of all possible errors:

```
2022    WRONG VALUE FOR PARAMETER NETCU
        Desired average number of reference points in CU negative,
        zero or too large ( > 90 ).

2035    TOO MANY POINTS IN LINE
        Number of points in a form-, break-, borderline exceeds the
        maximum (see table of restrictions).

2036    TOO MANY FORMLINES
        Number of lines exceeds maximum  (see table of restrictions).

2037    TOO MANY BREAKLINES
        Number of lines exceeds maximum  (see table of restrictions).

2038    TOO MANY BORDER/BREAK LINES
        Number of lines exceeds maximum  (see table of restrictions).

2039    TOO MANY BORDERLINES
        Number of lines exceeds maximum  (see table of restrictions).

2040    NOT ENOUGH POINTS IN DTM
        There are no reference points with height within the DTM
        limits.

2041    TOO MANY POINTS (LITE) OR PROTECTION KEY MISSING
        Too many points for SCOP-LITE Version or protection key
        missing or license file missing.

2050    NET-CU INDEX-BLOCK OVERFLOW
        Terrible large number of CUs ( > 5000000 )
        Check whether CU size too small or use smaller DEM limits.

2051    NET-CU INDEX-FILE TOO SMALL
        Internal error.
        Try another CU size; write ugly letter too the authors of
        this house of cards.

2052    NET-CU MASTER-INDEX TOO SMALL
        See 2050 and 2051.

2053    NET-CU GREATER THEN BLOCKLENGTH
        One CU contains more than 600 points.
        Try smaller CU size; check whether there are many points with
        the same coordinates (possibly due to an error during data
```

registration)

2054 TOO MANY LINES IN NET-CU
Number of lines within net CU is too high.

2059 INTERNAL ERROR (RXSTRN)
Could not find the number of a structure when computing line intersections.

2061 TOO SMALL WORKING STORAGE

Nasty error due to problems with memory allocation, should occur with small computers only (limited memory).
Try smaller CU size; buy larger computer.

2062 TOO SMALL WORKING STORAGE
See 2061.

2063 GROSS-CU TOO BIG
CU including overlap contains more than 600 points.
Cannot be recovered.
Use smaller CU size or smaller overlap.

2064 TOO MANY POINTS IN GROSS-CU
Can be recovered.
See 2063.

2065 TOO MANY LINES IN GROSS-CU
The sum of break lines and form lines in the CU plus overlap is greater than 50.
Use smaller CU size or smaller overlap.

2066 TOO MANY BORDERLINES IN GROSS-CU
Number of border line points in CU plus overlap exceeds 400.
Use smaller CU size or smaller overlap; do not use parameter DELIMIT.

2067 NO POINTS IN GROSS-CU
CU including overlap contains no points; this should happen with constant overlap only.
Use larger overlap or larger CU size; if then one of (2063-2066) occurs, go to 2051.

2068 WRONG CU-INDEX (INTERNAL ERR.)
Congratulations! You are the first who gets this message.

2083 TOO MANY GAPS IN CU-STRIP
In a CU strip from left to right a maximum of 10 gaps is allowed when using parameter DELIMIT.
Use larger CU size, omit parameter DELIMIT or use smaller LIMITs.

2084 See 2083.

2091 INTERNAL ERROR (STRIND)
DAF not in order; rerun SCOP.DMS to build new DAF.

2123 INTERNAL ERROR (RAENDL)
Try without parameter DELIMIT.

3012 NO SOLUTION FOR EQUATION SYSTEM
Too many points in one interpolation area.

3101 **WARNING: SIZE OF FILTER VALUES ?
Filter value given for bulk points smaller than filter values for other point classes; this seems to be unusual but it works.

3102 **WARNING: SIZE OF MAX FILT VALUES?
Maximum filter value given for bulk points smaller than maximum filter values for other point classes.

3103 **WARNING: SIZE OF CHECK VALUES ?
Check value for the bulk points in parameter CHECKCUR or CHECKLIN smaller then check value for other point classes.

3104 **WARNING: TOO MANY GRID INTERSECT.
An internal limitation is hit when storing the intersections between grid lines and border, break, form lines.
Not all lines are stored into the DTM. Computation continues but results may be affected.
For a new computation, a smaller CU size should be used; border, break and form lines should be checked.

3105 TOO MANY SECTIONS IN CU
A CU may be divided by break lines into a maximum of 127 interpolation areas. In this case, there seems to be a CU with more than six break lines which are almost all crossing each other.
Computation continues but result may be affected.
Check your break lines; use smaller CU size.

3106 **WARNING: BREAKL-DENSIF NOT POSS.
The densification points along break and form lines can not all be stored due to internal limitations (max. 800 points per CU). Densification stops for this CU; computation continues but result may be affected.
Use larger densification interval or smaller CU size.

3107 TOO MANY BREAK LINES IN CU
There are too many break lines in one CU; the subdivision of the CU into interpolation areas does not work.
Computation continues but result may be affected.
Check your break lines; use smaller CU size.

3108 **WARNING: TOO MANY BREAK LI POINTS

```
        Not all break line points can be stored when subdividing the
        CU into interpolation areas.
        Computation continues but result may be affected.
        Check break lines; use smaller CU size.

3121    **WARNING: TOO MANY POINTS IN CU
        With directive FAST, break and form lines are densified
        depending on grid width; a maximum of 1200 points per CU may
        be stored otherwise densification stops and this error
        message is printed.
        Computation continues but result may be affected.
        Check break and form lines; omit parameter OCTAND; use
        smaller CU size; use larger grid width; use another
        programme; do not use a computer.
```

12.4. Models derived by Hierarchic Robust Filtering

In this section the GUI for the hierarchic robust filtering will be described. Its theory is presented in section 20, general information on model derivation can be found in section 12.3. The aim of the hierarchic robust filtering is the elimination of gross errors from a given data set and the derivation of a model which is free of these blunders. For this aim data pyramides – similar to image pyramides – can be used to generate a surface model in a coarse to fine approach. Additionally, assumptions on the distribution of the gross errors can be considered. A typical application is the removal of off-terrain points in airborne laser scanner data (=ALS, also known as LiDAR=light detection and ranging). As a supplementary result, the input point data may be categorised into several classes in a finishing step.

In the hierarchic robust filtering a sequence of steps is applied. Each step can be seen as a transformation of data: one or two data sets (i.e. files) are taken as input, the data is processed and the result (up to three files; the classify step may produce even more) are the output. Input and output can be point cloud files and DTMs. The sequence of steps is called *strategy*, or more precisely a *filter strategy*. For different data sets, different strategies have been proven to be useful. There are eight different types of steps in the strategy, which will be described shortly. The output of one step serves as the input of the next step.

EliminateBuildings In an eliminate buildings step, the original project data or the output of the previous eliminate buildings step is processed. The input point cloud is split into building points (*step*i_*elb.(b)wnp*) and the output i.e. ground points (*step*i_*elg.(b)wnp*) (see Fig. 12.31).

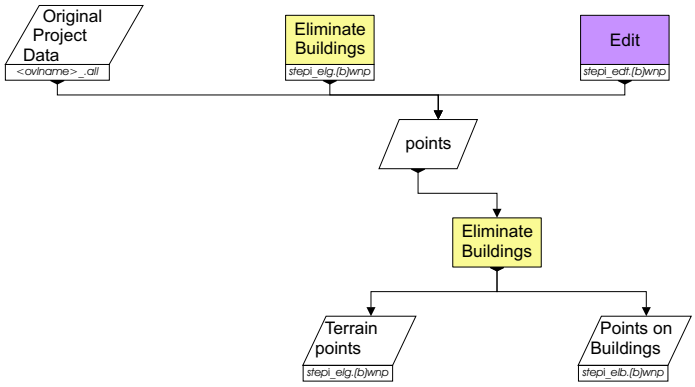


Figure 12.31.: Input and output of an EliminateBuildings step.

ThinOut In a thin out step a set of points is reduced in its details, the data is thinned out. This is performed on a raster basis. This step is useful for guaranteeing that a good mixture of blunders and terrain points is given. The input for this step is a set of points and the output (*step*i_*tho.(b)wnp*) is the reduced set of points (see Fig. 12.32).

Filter In a filter step a model is derived from a set of points which contains gross errors. These gross errors are also called off-terrain points. The aim is to eliminate these gross errors completely and build a ground model with the remaining points.

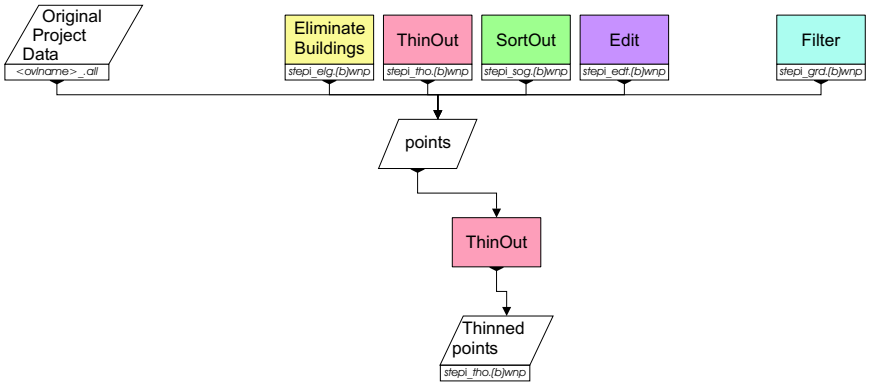


Figure 12.32.: Input and output of a ThinOut step.

This is possible, if there is a good 'mixture' of grossly false points and points free of these errors. The output is a file with the gross error points (*step*i*_veg.(b)wnp*), a file with the 'good' points (*step*i*_grd.(b)wnp*) and the DTM (*step*i*.dtm*) which is computed during the filter process (see Fig. 12.33).

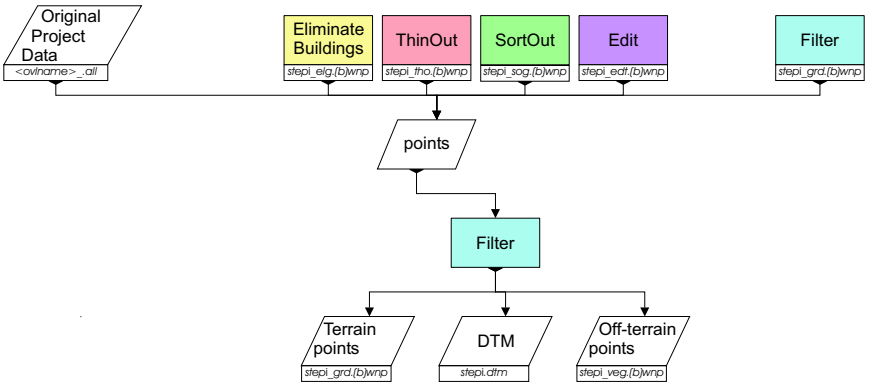


Figure 12.33.: Input and output of a Filter step.

Interpolate In an interpolate step a model is derived from the current data set by linear prediction (see section 12.3.7). The difference to a Filter step is, that no gross error detection is possible. The output is a DTM *step*i*.dtm* (see Fig. 12.34).

SortOut In a sort out step points are compared to a DTM. Therefore, the input are two files, a point cloud and a DTM. If the points are within a certain distance from the DTM they are accepted, otherwise they are rejected. The accepted points form the output file (*step*i*_sog.(b)wnp*) of this step (see Fig. 12.35).

Classify The classify step is a major extension of the sort out step. Likewise, points are compared to a DTM, but with respect to potentially more height difference intervals. Again, a point cloud and a DTM form the input, but the output may be one or more files, according to user specifications (see Fig. 12.36).

FillVoidAreas In a fill void areas step, the output of the previous step is processed (or the output of the last but one step, if the previous step is an interpolate step). Void areas in this point cloud are filled, the output is called *step*i*_fil.(b)wnp* (see Fig. 12.37).

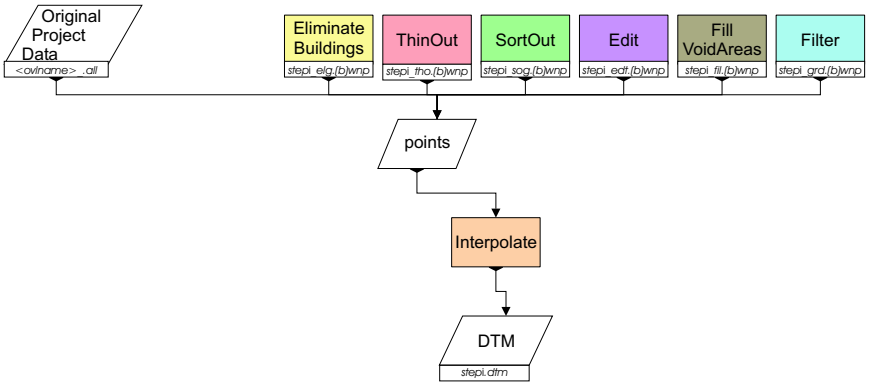


Figure 12.34.: Input and output of an Interpolate step.

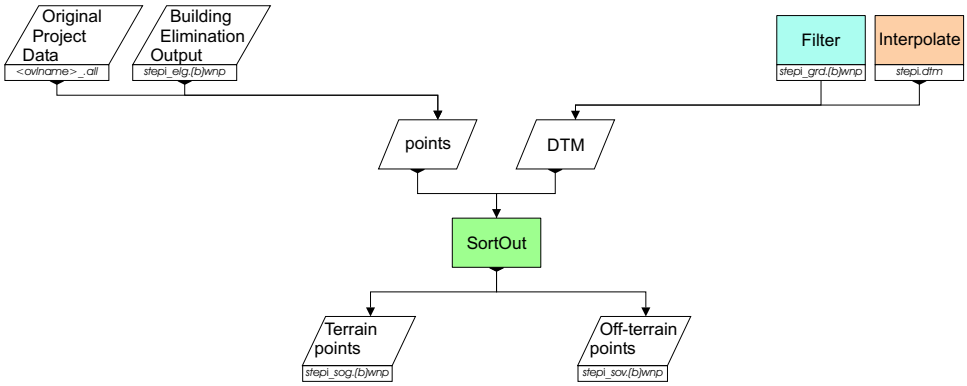


Figure 12.35.: Input and output of a SortOut step.

Edit The edit step allows the user an editing of the current data set (e.g. manual elimination of off-terrain points). The output of this edit process is a file (*stepi_edt.(b)wnp*) containing the manipulated point data (see Fig. 12.38).

In the dialog window *Derivation of the DTM* the radio button *Robust Filtering* has to be chosen in order to apply the hierarchic robust interpolation (see Fig. 12.20). On pressing the button *Details Robust*, the *Robust filter strategy* window appears (see Fig. 12.39). Within this window the composition of new strategies and the retrieval of existing strategies is possible. As described in section 12.3.5 it is possible to store and retrieve strategies with the group *Parameter persistence*. In the middle of the window the current strategy is presented. The steps are sorted from the first (Step 0) to the last from top to bottom. The strategy in the example consists of two steps only, a ThinOut step which is followed by a Filter Step. In the left area of the window the tools for editing a strategy are placed. A strategy is built step by step using the *Modify strategy* operations. At a certain position a step can be inserted or removed using the button *Realize*. For the insertion the type of the step must be specified by a radio button of the group *Add*.

Not every combination of steps yields a valid strategy. The rules for a valid strategy can be found in section 12.4.9.

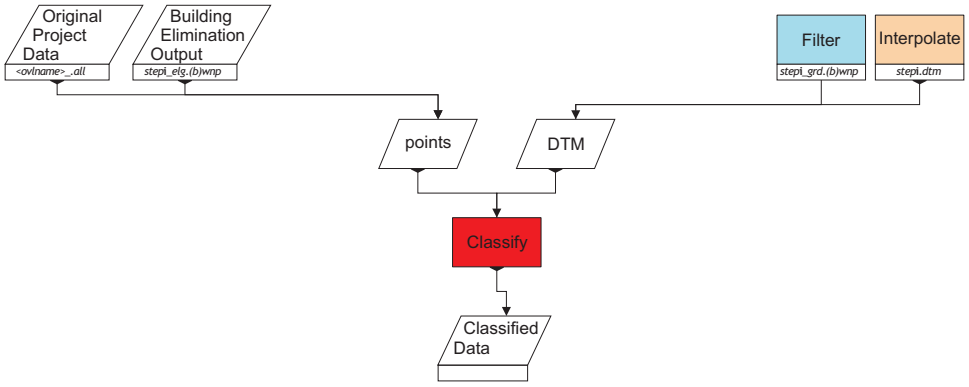


Figure 12.36.: Input and output of a Classify step.

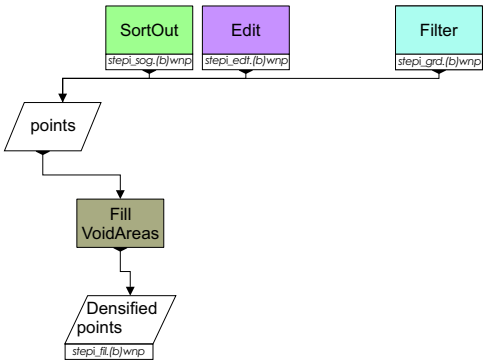


Figure 12.37.: Input and output of a FillVoidAreas step.

While the strategy is built step by step the sequence of the steps is not checked. If, however, an incorrect strategy is accepted by pressing button *OK* or button *Apply*, the user will be notified of the first error (impossible sequence) in the strategy.

Each step on the GUI is represented by a *State-Switch* (see section 5.1.1) which allows to *Activate* or *Deactivate* a step during the model computation. Additionally a *Properties...* window allows to specify the parameters for each step.

In the following the Filter step will be described first, followed by the description of EliminateBuildings, ThinOut, Interpolate, SortOut, Classify, FillVoidAreas and Edit. The Filter can be considered as the 'heart' of the robust filter algorithm. If a thorough mixture of ground points and off-terrain points is given from the beginning, the application of one Filter step can be sufficient.

12.4.1. Filter Step

Right-clicking on *Filter Step i...* opens the dialog window *Filter Step i* (see Fig. 12.40). In robust filtering the model is derived in an iterative fashion. First, all the bulk points have the same weight. As linear prediction is applied for the surface computation (see section 12.3.7) the points will either lie on, above or below the computed surface, according

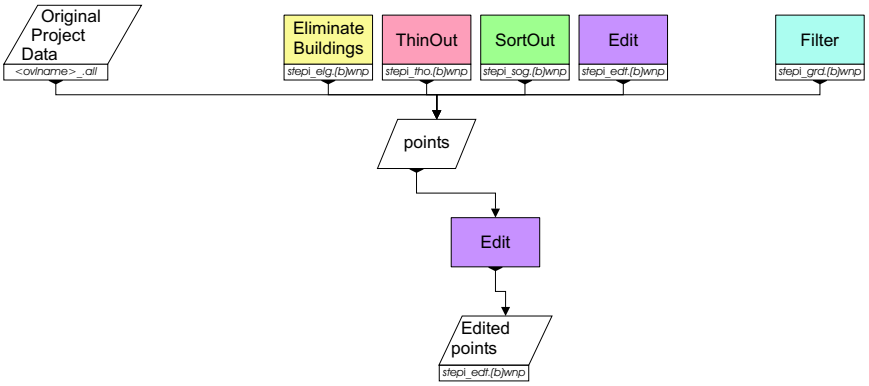


Figure 12.38.: Input and output of an Edit step.

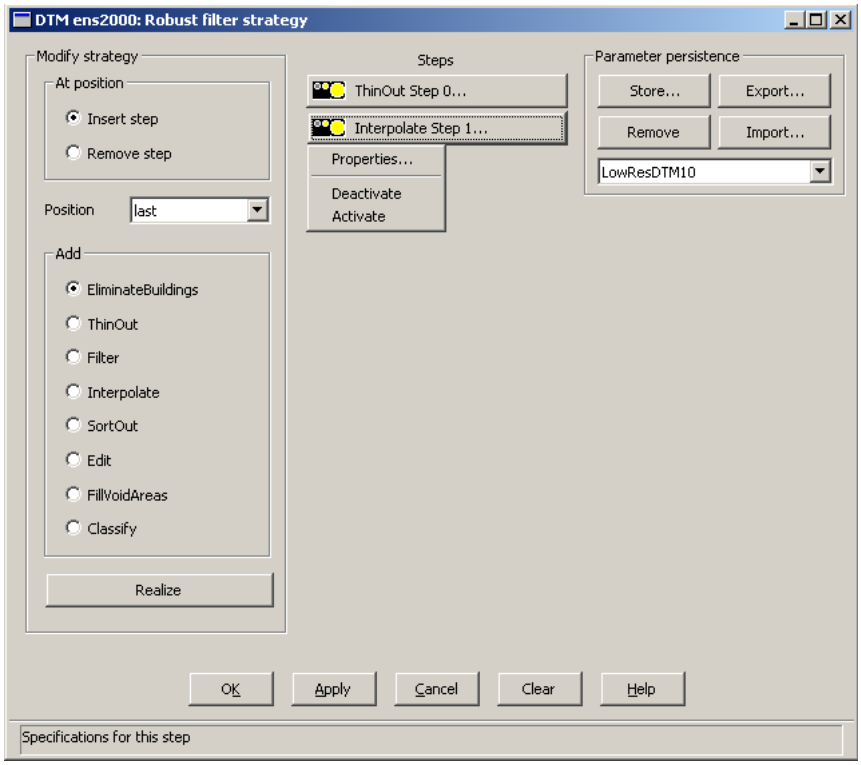


Figure 12.39.: The window *Robust Filter Strategy*

to the specified stochastic model[2]. After the first model derivation the filter values (i.e. the residuals) are computed and the weights of the points are altered according to the *weight function*, which is shown in the upper middle of the Filter window. If a point is given a low weight, it will have little influence on the run of the surface in the next iteration. In the shown example this is the case for points lying more than 1 meter (unit of z-values) above the terrain, which can be seen at the graph of the weight function. If it has full weight, in the example this is the case for all points which are on or below

[2] A second method for model derivation is available, too, which also has the property to approximate the points rather than strictly interpolating them. It will be described later in the text.

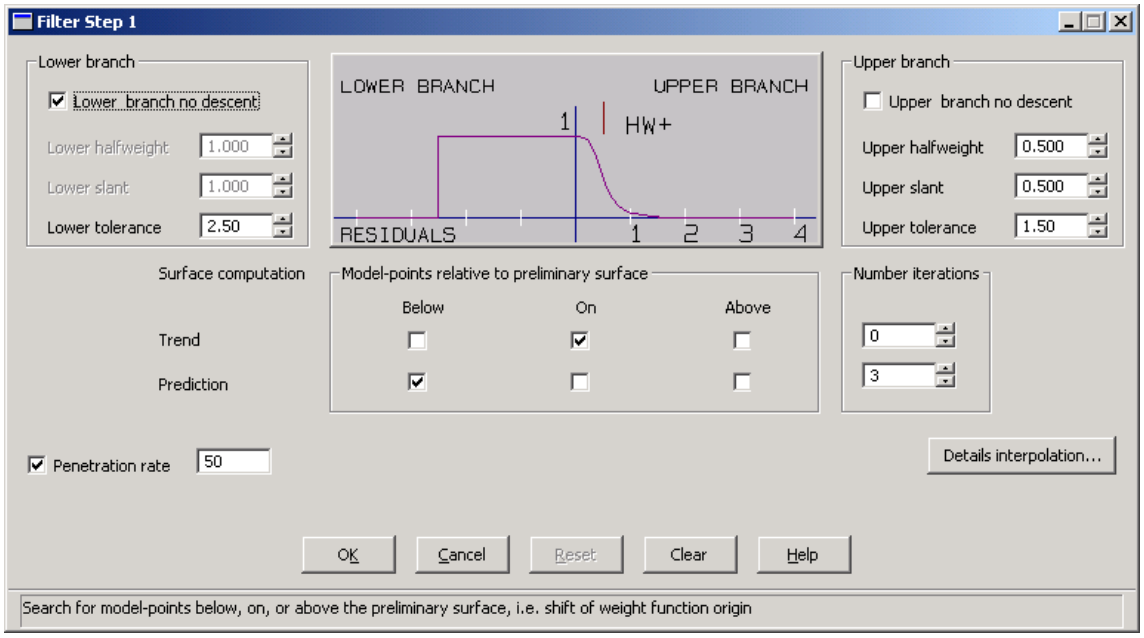


Figure 12.40.: The window *Filter Step*

(until the *lower tolerance* value is reached) the preliminary surface, the surface is attracted strongly to this point. The maximum of the weight function is 1, which is always the value of the origin of the weight function. The minimum value is zero.

The weight function is not symmetric to its origin. It can be specified in the group *Lower branch* and the group *Upper branch* which are left and right, respectively, of the graph of the weight function. With checkbox *Lower branch no descent* and checkbox *Upper branch no descent* the lower and the upper branch, respectively, are set to a straight line. This means, that the weight function does not descent to zero but keeps the maximum value of one. Otherwise, i.e. if checkbox *branch no descent* is not checked, the weight function can be specified by its two parameters *Halfwidth* and *Slant*, which are specified in the corresponding numeric fields. Again, this is possible independently for the lower and the upper branch. An usual default is to set the Slant and the Halfwidth to equal values. If a filter value (i.e. the distance of a z-measurement to the interpolated terrain surface) has the size of Halfwidth, it means that the weight will only be 1/2 for the next iteration. With Slant the steepness of the weight function can be controlled at the Halfwidth value. The specified values are always positive, but those for the lower branch are multiplied internally with –1. In the graph the values of the Halfwidths are drawn and annotated with HW+ and HW–. Additionally, the weight function can be set to zero above and below a certain threshold. In the shown example these thresholds are –2.5 and +1.5 units. Again, for the lower cut-off value of the weight function only its absolute value is specified. The shape of the graph of the weight function is changed according to the values of Halfwidth, Slant, and Tolerance. The weight function is used to specify which weights are given to points for the next iteration in the model derivation.

A control is necessary which tells the process, if it is end-iterated, or not. In principle, this is performed internally. The algorithm can determine, if gross errors are still in the data or not. However, it is advisable to specify a maximum number of iterations in order to

keep computation time short and also to handle pathological cases, where an end-iterated status may be reached only after very many iterations. Thus, it is possible to specify the maximum number of iterations in a Filter step. As it can be seen in Fig. 12.40 there are two possibilities to specify this number (group *Number iterations*). To explain this, it will be necessary to take a closer look on how the surface is computed.

There are two methods to compute the surface which can be used in a Filter step. Both are embedded in a re-computation framework based on the filter values and the weight function as described above. The two methods are:

Prediction This method is described in section 12.3.7 and 12.3.8. It is flexible in the sense that the computed surface approximates the given data and follows the local minima and maxima and different slopes. Furthermore, the (assumed) accuracy of the points can be specified as a-priori filter value. The flexibility is controlled by the covariance function which is determined from the data. If too many gross errors are in the data, this determination is prone to fail and the surface will run through all points, ground points as well as blunders.

Trend This method computes for each computing unit one plane. A plane is very inflexible, but this can be an advantage. The flexibility does not depend on the number of gross errors. Theoretically, the portion of gross errors can be larger than 50% and a correct removal of gross errors is still possible, at least if their distribution is known approximately.

This description suggests to apply iterations with a Trend as surface computation method first which is then followed by a Linear Prediction surface to gain the necessary flexibility. Exactly this is possible in a Filter step.

The maximum number of iterations with a plane (upper numerical field) and with the predicted surface (lower numerical field) is specified in the group *Number iterations*. However, if many break lines are in the terrain, which are only sampled by randomly distributed laser scanner points, it is better to use only Linear Prediction as surface computation method. Otherwise, the break lines can be rounded off strongly.

So far, the methods for the surface computation and the weight function have been described. However, one aspect of the weight function remains to be specified, and this is the origin of the weight function. The origin, i.e. the value for which always the weight 1 is returned, is not necessarily on the zero point of the real axis. It can also be shifted relative to the preliminary surface. A shift to the lower is typical for airborne laser scanner data[3]. This shift is determined automatically from the filter values. A detailed description of the methods can be found in section 20. However, it is left to the user to specify, if such a shift in the data is likely or not. In the group *Model-points relative to preliminary surface* this can be specified with checkbox *Below, On, and Above* independently for computation with the Trend and Prediction surface model. If checkbox *On* is chosen, the origin is always the zero point of the real axis, if checkbox *Below* or checkbox *Above* are specified, the optimal place for the origin is searched below or above the preliminary surface, respectively. Finally, an additional method to find this shift for airborne laser scanner data is the specification of an estimated penetration rate[4]. This is accomplished by checking

[3] This indicates, that all the ground points have negative filter values (lie below the preliminary surface). This is the case, if the surface runs in an averaging way between the ground points and the off-terrain points. For laser scanner data this is typically the case.

[4] The penetration portion of ground points in the given point set. It is specified in percent.

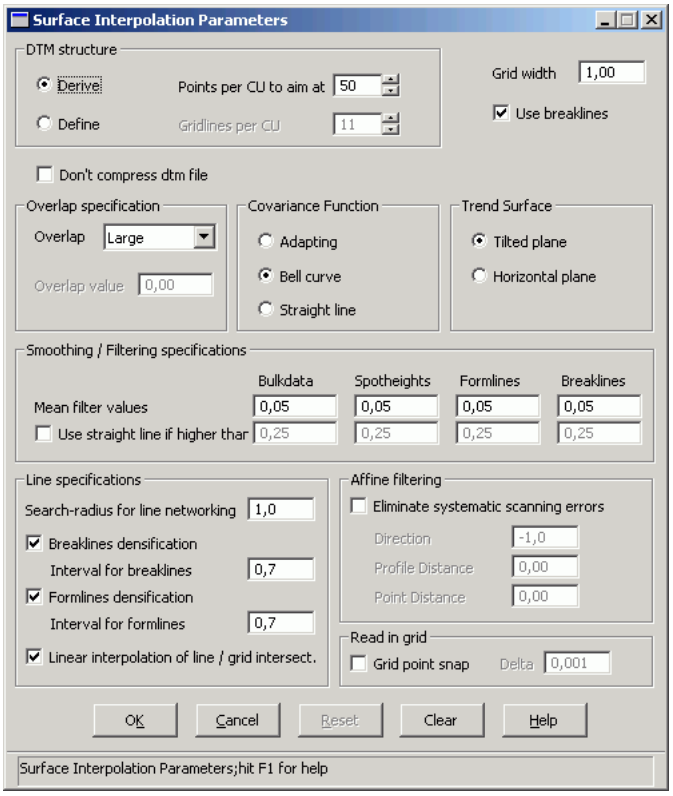


Figure 12.41.: The window *Surface Interpolation Parameters*

checkbox *Penetration rate* and entering the estimated value in the corresponding field. More precisely, this method is applicable, if mostly gross errors above the surface can be expected and an estimation of the number of gross errors is at hand.

The parameters of the linear prediction are specified in an own window which opens on pressing the button *Details interpolation...* (see Fig. 12.41). The parameters are the same as in section 12.3.7, but only those which make sense for the filtering of gross errors can be specified. Additionally the use of breaklines can be omitted.

12.4.2. EliminateBuildings Step

Right-clicking on *EliminateBuildings Step i...* opens the dialog window *EliminateBuildings Step i* (see Fig. 12.42). Within this step Lidar points on top of buildings are eliminated. The input file is sorted into small quadratic cells of size *Cell size*. The slope between neighbouring cells are computed. Groups of cells with low slopes surrounded by cells with slope $>=$ *Minimal slope* are assumed to be a building, if the group of cells form an area of size $>=$ *Minimal area*.

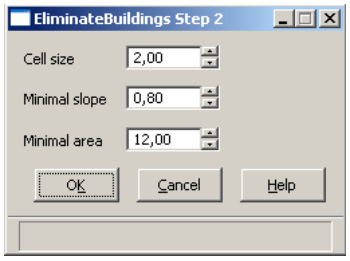


Figure 12.42.: The window *EliminateBuildings Step*

Useful values for these parameters are:

Cell size	2 meter
Minimal slope	0.9 (tan)
Minimal area	$20 - 60$ meter^2

The program works best with high point densities and when both first and last pulse points are in the input file.

12.4.3. ThinOut Step

Right-clicking on *ThinOut Step i...* opens the dialog window *ThinOut Step i* (see Fig. 12.43).

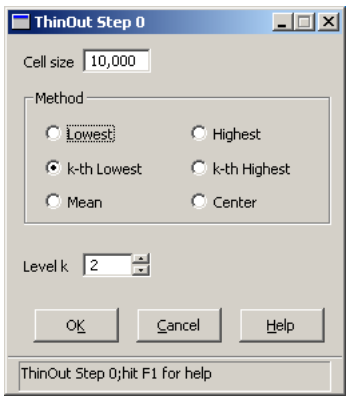


Figure 12.43.: The window *ThinOut Step*

A raster is laid over the complete data set and one point per cell is chosen for the output file, which is the thinned out 'version' of the input file. The cell width – it is quadratic – is specified in the numeric field *Gridsize*. Of course, the application makes only sense, if the parameter for the gridsize is larger than the minimum point distance. To gain a notable effect, the parameter has to be twice the size of the minimum point distance. For airborne laser scanner data or other methods which generate very dense point sets this value should be specified as a multiple of the average point distance. If the density varies strongly, which can be the case for topographic terrestrial laser scanner data, it can be thought of a minimum point distance, which should be achieved in the thinned out data set.

There are different possibilities to compute the output point for each cell. The ones which are available in a ThinOut Step are:

Lowest In each cell the point with the lowest height is chosen as the output point.

Highest In each cell the point with the largest height is chosen as the output point.

k-th Lowest In each cell the point with the k-th lowest height is chosen as the output point. The level of *k* is specified in the numeric field *Level k*.

k-th Highest In each cell the point with the k-th largest height is chosen as the output point. The level of *k* is specified in the numeric field *Level k*.

Mean In each cell the mean of all points within the cell (center of gravity) is chosen as the output point.

Center In each cell the point is chosen as output which has the smallest planimetric distance from the cell middle.

If long-range-observations are expected in the data set, it is often preferable to use *k-th Lowest* instead of *Lowest*. The same applies for *Highest/k-th Highest* if short ranges are expected. Further comments on the choice of the method can be found in section 20. It shall be noted that this is a simple data reduction scheme which does not aim at preserving topographic features.

12.4.4. Interpolate Step

Right-clicking on *Interpolate Step i...* opens the dialog window *Interpolate Step i*. For the interpolation without using individual weights it is only necessary to specify the parameters of the linear prediction. Therefore the same window as in the Filter step is used (see Fig. 12.41). It can be understood as filter where all points are classified as terrain points.

12.4.5. SortOut Step

Right-clicking on *SortOut Step i ...* opens the dialog window *SortOut Step i* (see Fig. 12.44). For a SortOut step the input consists of two data sets. One is a DTM and one is a set

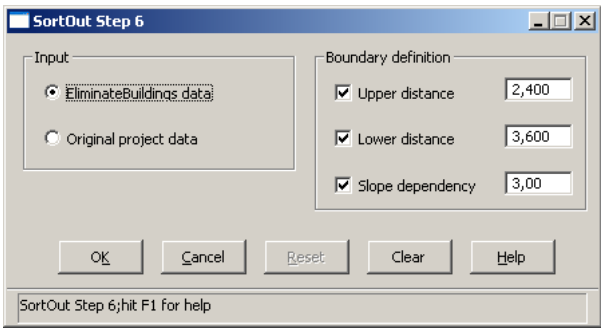


Figure 12.44.: The window *SortOut Step*

of points. Users may select to process the output of the last EliminateBuildings step (*EliminateBuildings data*) or to use the *Original project data*. If *EliminateBuildings data* is selected, but no such step exists, the *Original project data* will be employed. Each point is compared to the DTM and the z-difference is computed. If the z-difference is within a certain interval, then the point will be in the output data set. Primarily, the interval

is specified by *Upper distance* and *Lower distance*. Having activated the check box, the respective interval border can be specified as absolute value. Otherwise, plus or minus infinity is used. *Slope dependency* may be used to apply an additional, dynamic widening of the admissible interval, considering the local DTM slope. The actual interval borders are then adapted according to:

$$b_{actual} = b_{set} + b_{set} \cdot sloDep \cdot slope \tag{12.1}$$

Where b_{actual} denotes the interval border actually used, b_{set} represents one of the interval borders set in *Upper distance* and *Lower distance*, $sloDep$ stands for the numerical value set in *Slope dependency*, and $slope$ is the local slope of the DTM.

12.4.6. Classify Step

Right-clicking on *Classify Step i ...* opens the dialog window *Classify Step i* (see Fig. 12.45). Being a strategy's last step, the Classify step categorizes the overlay data into disjoint sets

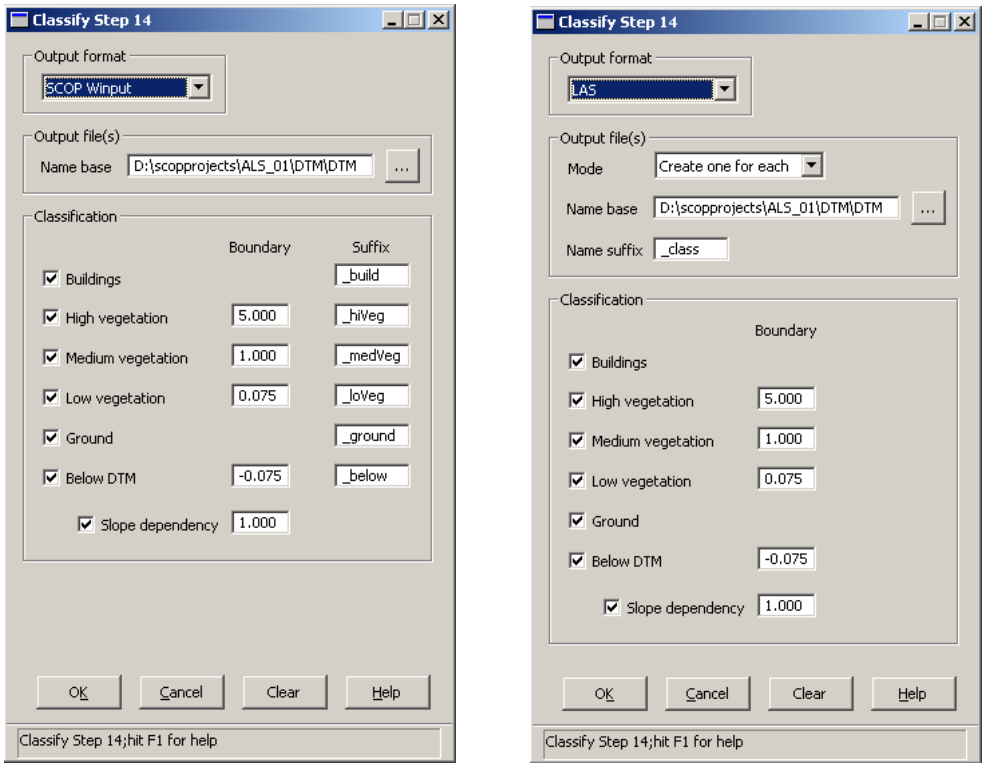


Figure 12.45.: The window *Classify Step* with two different layouts, depending on the output format: (binary) Scop Winput and (binary) XYZ (left), or LAS (right)

based on the final DTM and the output of eventual EliminateBuildings steps. All points are assigned to one of the following classes, depending on user specifications:

- *Buildings*
- *High vegetation*
- *Medium vegetation*

- *Low vegetation*
- *Ground*
- *Below DTM*
- *Unclassifiable*
- *NotToBeClassified*
- *Others*

Points defined in input LAS-files may already hold edited classification flags. Thus, for the editing of LAS-files (Output format LAS / Modify input files) there exist some special rules that are marked with (LasEdit) in the following.
A Classify step processes each input point separately until being assigned to a class, in the following sequence:

1. Structure data inside the model limits are given the class *Ground*.
2. Data outside the Model limits and echoes other than the imported one (LasEdit) are classified as *NotToBeClassified*, unless they already hold an edited classification flag (LasEdit); these points keep their old classifications and appear in the protocol as belonging to *Others*.
3. Points ocurring in the respective output file of any EliminateBuildings step are given the class *Buildings*.
4. Points with height differences larger than the boundary of *High vegetation* are assigned to this class.
5. Data featuring height differences between *High vegetation* and *Medium vegetation* are assigned to the class *Medium vegetation*.
6. Data with height differences between the boundaries of *Medium vegetation* and *Low vegetation* are given the class *Low vegetation*.
7. Data featuring height differences smaller than the boundary of *Below DTM* are assigned to the class *Below DTM*.
8. The remaining points are given the class *Ground*.

The vegetation classes and the class *Below DTM* are defined by adaptive minimum (*Below DTM*: maximum) height differences between the data and the DTM. Like in the SortOut step, the class boundaries may be widened according to local DTM slope by means of the checkbox *Slope dependency* and the adjoining numeric field. The actual boundary of class *BelowDTM* is computed point-wise through application of formula 12.1 to the boundary of *BelowDTM*, while the vegetation classes' boundaries are all shifted by the same value resulting from the application of the formula to the lowest active vegetation class boundary.

The determination of DTM height differences fails at planimetric positions without valid DTM heights, e.g. originating from a locally sparse data distribution or due to being located at the margin of the input data. The respective data are assigned to the class *Unclassifiable*, unless they already feature an edited classification flag (LasEdit).

Users may freely select a subset of the possible data classes for processing by deactivating the respective check boxes. If class *Buildings* is deactived, a Classify step will assign the

Table 12.7.: Mapping of Scop classes to LAS

Scop class name	LAS class name	LAS class bit
Buildings	Building	6
High vegetation	High vegetation	5
Medium vegetation	Medium vegetation	4
Low vegetation	Low vegetation	3
Ground	Ground	2
Below DTM	Low point (noise)	7
Unclassifiable	Unclassified	1
NotToBeClassified	Created, never classified	0
Others	<all others>	

output points of eventual EliminateBuildings steps to other, active classes. When deactivating a vegetation class, the respective points will be classified as belonging to the next lower active vegetation class, or the class of ground points, respectively. Analogously, when deactivating the classification of *Below DTM*, the respective points will be classified as belonging to *Ground*. Deactivation of the class *Ground* results in the respective points being classified as *NotToBeClassified*.

Five output file formats are provided: (binary) Scop Winput, (binary) XYZ, and LAS. As XYZ and Winput do not feature the storage of point classifications, for these formats, the classified point sets are directed to separate files. The according file names are constructed from the path and prefix specified in the text field *Name base*, and from the suffices defined for each class in the text fields of column *Suffix*. Several classes may share the same output file, for instance all data above and below the DTM. Alternatively, data may be classified, but not output. This is achieved by clearing the suffix of the respective class. By e.g. activating the class *Below DTM* and clearing the related file name suffix, data below the DTM are separated from *Ground* data, but not exported to file. Data resulting as *Unclassifiable* is written to a file whose suffix cannot be altered by users. Its name is <*Name base*>_unclassifiable.<ext> .

Unlike the other output file formats provided, LAS facilitates the storage of point classifications, and the output of classification may be directed to a single file. Thus, in the LAS layout there is just one file name suffix. However, for LAS output there is another option available concerning the storage of results in the drop down list *Mode*: *Create one for all* yields a single LAS file with classified points from all input files, *Create one for each* results in one file of classified points per input file, and *Modify input files* edits the classification flags in the original input LAS files without changing them in any other way. The last two options are only available in file-based overlays (cf. section 5.3.11). Table 12.7 presents the mapping of Scop classes to their equivalent representation in the LAS format.

12.4.7. FillVoidAreas Step

Right-clicking on *FillVoidAreas Step i...* opens the dialog window *FillVoidAreas Step i* (see Fig. 12.46). During robust filtering, quite large areas without ground points may appear,

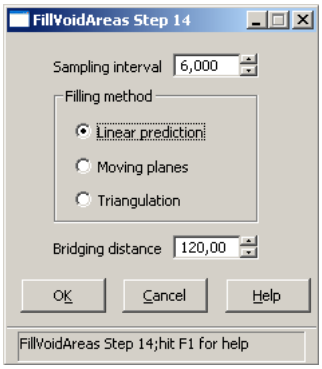


Figure 12.46.: The window *FillVoidAreas Step*

which can get a problem for DTM interpolation. A FillVoidAreas step solves this deficiency by filling these gaps. New filling points are inserted on a regular grid, having the distance of *Sampling interval* from each other. Points will be inserted only at positions where the distance to the next data point exceeds the *Sampling interval* and is smaller than the *Bridging distance*. The method to estimate filling point heights can be selected in the group *Filling method*:

- *Linear prediction* results in a smooth, undulating run of filling point heights
- *Moving planes* is the fastest method, but may yield a rough surface
- *Triangulation* produces a waveless, but jagged run of heights

12.4.8. Edit Step

Right-clicking on *Edit Step i...* opens the dialog window *Edit Step i* window (see Fig. 12.47). With the help of this window an appropriate edit tool can be specified. Using such an Edit step allows the user to stop the automatic processing and inspect and manipulate the intermediate classified data.

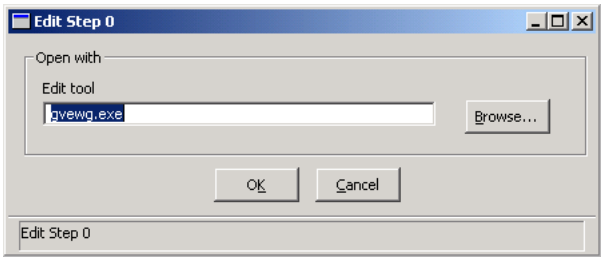


Figure 12.47.: The window *Edit Step*

12.4.9. Assembling of Steps

So far, the single steps of a filter strategy have been described. It remains to be explained, which combination of steps leads to a valid strategy. The rules for valid strategies are:

1. The first step must not be a SortOut, Classify, or FillVoidAreas step.
2. The last step must be a Filter, Interpolate, SortOut, or Classify step.
3. EliminateBuildings steps must be performed as the beginning sequence, eventually interrupted by Edit steps. This sequence must be followed by a ThinOut, Filter, Interpolate or Edit step.
4. ThinOut can only be performed after Edit, Filter, SortOut, EliminateBuildings or at the beginning.
5. SortOut can only be performed after Edit, Filter or Interpolate.
6. Filter can be performed after all steps, at the beginning and at the end.
7. Interpolate can be performed after all other steps (i.e. not Interpolate), at the beginning and at the end.
8. FillVoidAreas may only be placed within strategies in two ways: FillVoidAreas-Interpolate-End, or FillVoidAreas-Interpolate-Classify-End.
9. Classify can only reside at the end of a strategy (possibly followed by Edit, however).
10. Edit can be performed after all steps and at the beginning.

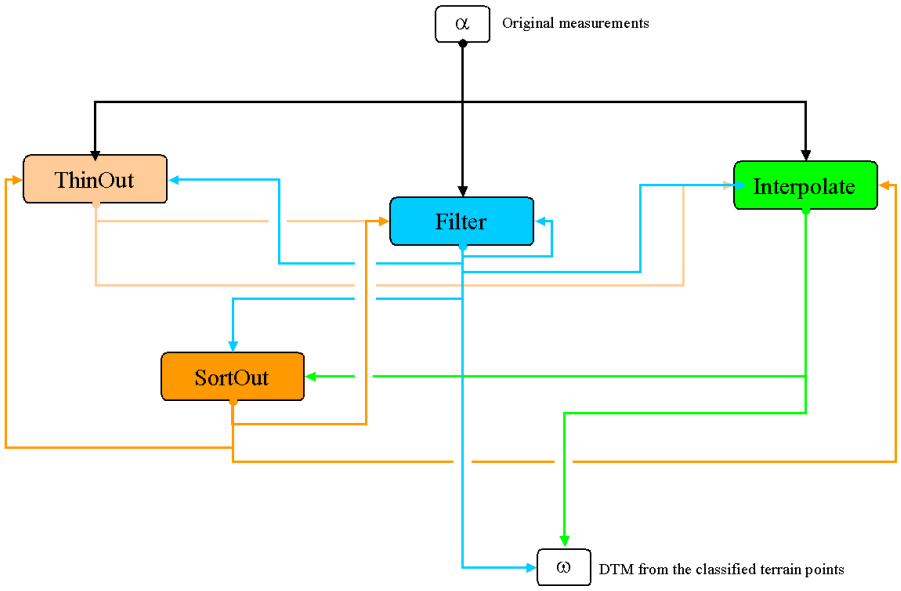


Figure 12.48.: Sequence of steps within a strategy for the hierarchic robust filtering.

The sequence of all possible steps (due to simplification the EliminateBuildings, FillVoidAreas, Classify and Edit steps are not mentioned) within a strategy starting from the original measurements to the final DTM computed with the accepted terrain points can be seen in Fig. 12.48. The arrows of the steps point to all possible subsequent step types. The first step (either Eliminate Buildings, ThinOut, Edit, Filter or Interpolate) always takes the original measurements (i.e. a cloud of points and possibly some line information) as input. The output is a file with point information which will be the input for the subsequent step and a DTM (for Filter and Interpolate). In general the output of one step is used as the input for another step. After the last step of a strategy, which must

always be a Filter or Interpolate step, the resulting DTM is the final model. If a SortOut follows a Filter or Interpolate step, the DTM is used as input for the SortOut step. The second input to SortOut is either the output of the last Eliminate Buildings step, or the file with the original measurements. The output of SortOut can be the input for all other steps. An Edit step can always be inserted if data manipulation before or after a step is necessary. Graphically, and much clearer, the Input and Output of the step types within a strategy is shown in Fig. 12.49. For simplification purposes the EliminateBuildings step, FillVoidAreas, Classify, and the Edit step, which can be used for point manipulation of the current data, are not included. The steps (i.e. the computation processes) are rectangles with a light grey background. As mentioned above, SortOut cannot be the first step, because no DTM is available at the beginning. Filter or Interpolate, eventually followed by a SortOut, must be the last steps because the generation of a DTM from the classified ground points is the ultimate aim. In the figure blue arrows indicate output, orange arrows indicate input. Different line types indicate the input for a Filter, Interpolate, SortOut and ThinOut step.

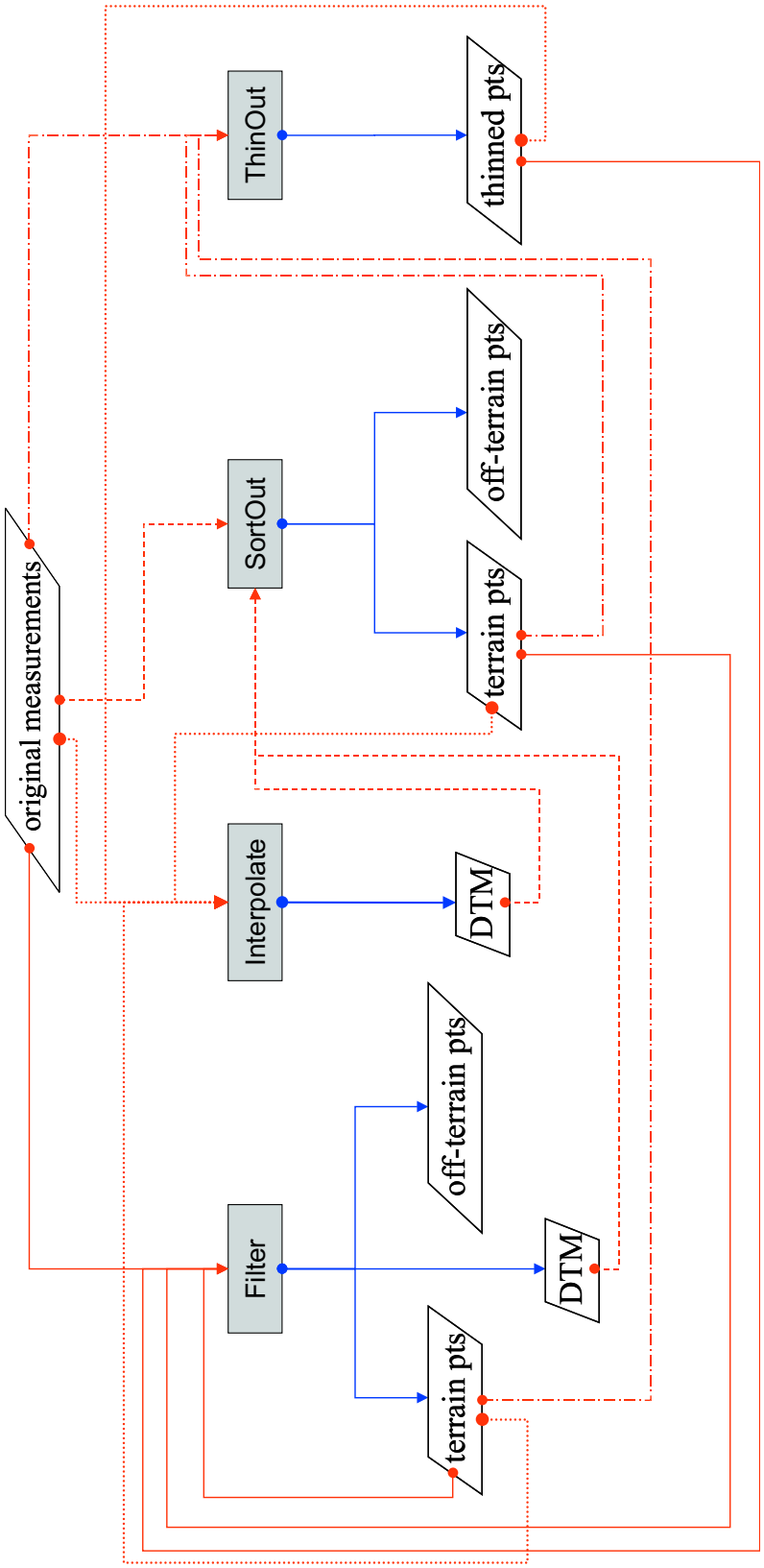


Figure 12.49.: Input and Output within a strategy for the hierarchic robust filtering.

A commandLine / commandFile (cmL/cmF) example for the generation of a simple hierarchic robust filter strategy is:

cmL example

```
@SimpleLaserStrat;
 dtm, model, Robustfiltering, DetailsRobust/ Clear,
// ThinOut Step 0
      InsertStep, ThinOut, Realize,
      ThinOutStep0/ CellSize=(10.00), Lowest, ok;
// Filter Step 1
      Filter, Realize,
      FilterStep1/ Clear,
        LowerBranchNoDescent 0 3.0 3.0 5.0,
        UpperBranchNoDescent 0 0.5 0.5 0.75,
        TrendBelow 1, PredictionOn 1,
        TrendNumberIterations 2 3,
        PenetrationRate 1 30,
// Details Interpolation
        DetailsInterpolation/ Clear,
          Gridwidth 5, Define 1 21,
          Overlap Large,
          FilterBulkdata 0.10 0.01 0.03 0.02,
          ok;
       ok;
// SortOut Step 2
      SortOut, Realize,
      sortoutstep2/ Clear,
        LowerDistance 1 10.00,
        UpperDistance 1 2.50,
      ok;
// Filter Step 3
      Filter, Realize,
      FilterStep3/ Clear,
        LowerBranchNoDescent 0 3.0 3.0 5.0,
        UpperBranchNoDescent 0 0.25 0.25 5.0,
        TrendOn 1, PredictionBelow 1,
        TrendNumberIterations 0 3,
        PenetrationRate 1 50,
// Details Interpolation
        DetailsInterpolation/ Clear,
          GridWidth 2, Define 1 11,
          Overlap Large,
          FilterBulkdata 0.10 0.01 0.03 0.02,
        ok;
      ok;
@endProc;
```

12.4.10. Default Values and Predefined Strategies

Currently, twelve predefined strategies can be chosen from the group *Parameter Persistence* (see above). These strategies will be described shortly.

Dependent Strategies

The parameters of these strategies depend on the *Basic settings* in the window *Derivation of the DTM*. These dependencies are the result of efforts to ease the creation of best-fitting parameter sets for a wide range of data conditions and filtering requests. Dependent strategies are indicated by an asterisk (*) at the end of their name. Currently, there are six dependent parameter sets available. 'Lidar Default*' and 'Lidar DTM Default*' aim at producing an optimum DTM in the majority of cases, while 'Lidar Default Strong*' classifies more points as off-terrain, and 'Lidar Default Weak*' keeps more points as terrain points. 'Lidar DSM*' produces a surface model with just one level of robust filtering. Manual alteration of such a strategy will cut its dependencies, the group *Basic settings* will get inactive and the name of the parameter set will jump to 'Unnamed parameter'.

Fix Strategies

LowResDTM10 This strategy is intended to obtain a surface model from a reduced set of points. The reduction is performed by taking the mean point within a 10m raster cell (ThinOut Step 0...).

SymmetricBlunders This strategy is an example for the elimination of gross errors above and below the terrain surface. This is performed in an iterative process (Prediction with 3 iterations) using a symmetric weight function. With the help of the lower and upper tolerance values a classification in terrain and off-terrain points is performed.

LidarSimpleFilter This strategy works with two data pyramid levels (10m and original data). In the first step (ThinOut Step 0...) the data set with the lower resolution is generated. In the second step (Filter Step 1...) gross error elimination is performed. Afterwards SortOut is used for accepting all original points within a certain tolerance to the coarse surface model. In the final step the surface model is generated using a further Filter step.

Rural(PtnDens3) This strategy uses three data pyramid levels (15m, 7m and original data) to obtain a digital terrain model from airborne laser scanner data in a coarse to fine process. The original data is assumed to have a linear point distance of approximately 3m. After the classification in each data pyramid level (with one to two Filter steps) the reference model for data densification is computed by an Interpolation step. In this step all classified terrain points are equally weighted. The final model is computed with a *Grid width* of 3m from the equally weighted classified terrain points.

Urban(PtnDens1) This strategy uses four data pyramid levels (2m, 5m, 2m, and original data) to obtain a digital terrain from airborne laser scanner data in a closely blocked area. Additionally points below the terrain surface (caused by *Long Ranges* in laser

scanner data) are preliminary eliminated in the fist two steps. In the following steps off-terrain points are eliminated by robust interpolation (Filter step) in a course to fine process with the help of data pyramides. The final surface model is computed in step 13 by an Interpolate step with equally weighted terrain points. The original data is assumed to have a linear point distance of approximately 1m.

Rural(PtnDens1) This strategy uses three data pyramid levels (10m, 5m and original data) to obtain a digital terrain model from airborne laser scanner data in a wooded area. This strategy is similar to the Urban(PtnDens1) strategy, but uses different data pyramid levels and thresholds. A pre-filtering of points below the terrain surface is not performed. The original data is assumed to have a linear point distance of approximately 1m.

12.5. Data

In each *Model Overlay* window (see an example in Fig. 12.1) there is a group of state-switches below the label *Input Data* representing all kind of data belonging to this model. Additionally there is a state-switch *Data* .... This state-switch has three states:

Off: Data is not visible in the main graphics panel.

Display: Data is visible in the main graphics panel but editing is prohibited.

Edit: Data is visible in the main graphics panel and editing is possible.

Clicking at the pop-up menu *Properties* of this state-switch opens a the window *Data Properties of ens2000* (see Fig. 12.50) where several settings can be made for each type of data.

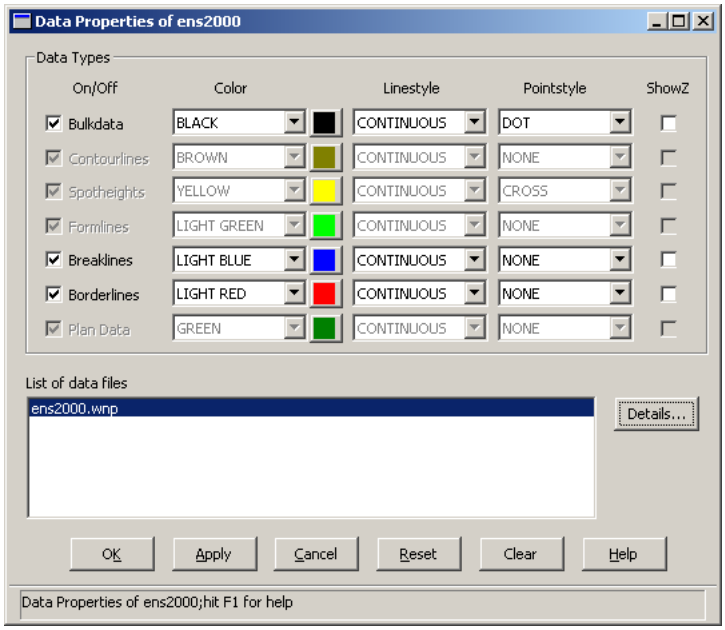


Figure 12.50.: The window *Data Properties*

First of all the visual appearance of each data type can be modified by selecting corresponding colors, line styles and point styles using the respective selections *Color, Line style and Point style*. Moreover each data type can be shown or hidden on the screen by selecting the appropriate checkbox under the label *On/Off*. This is particulary useful for bulk data, since a huge amount of data can severely decrease the system speed.

When clicking the checkbox *ShowZ* of each data type, the data points will be annotated with their heights in the main graphics panel which can be useful for error detection.

A list of all data files actually imported to the model overlay is shown in the selection *List of data files*. To view some statistical information about a certain data file select the file name from the list of available data files and press button *Details*.

The minimum and maximum coordinates as well as the list of all contained feature codes are displayed in a separate window. To close this window press button *Close*.

To import data to or to remove data from the overlay use the appropriate dialogs (see sections 12.2.1 and 12.2.2for details).

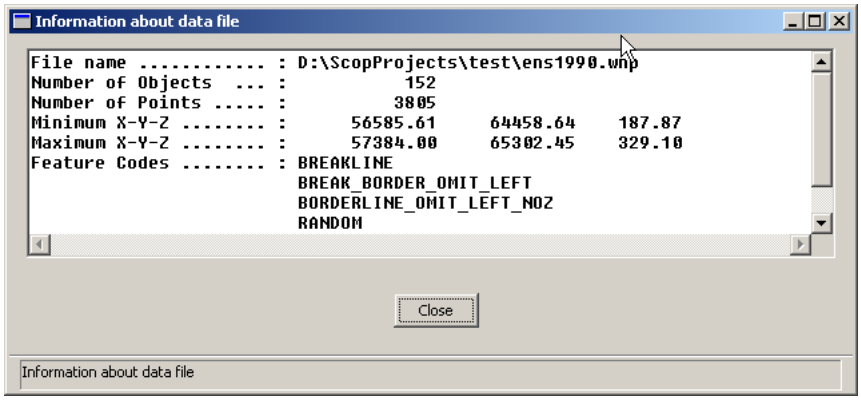


Figure 12.51.: The window *Information about data file window*

12.6. Isolines

This chapter describes the definition and manipulation of the parameters for the computation of isolines. In the following we denote with the term isolines lines of equal Z coordinate. In digital elevation models isolines are termed contourlines.

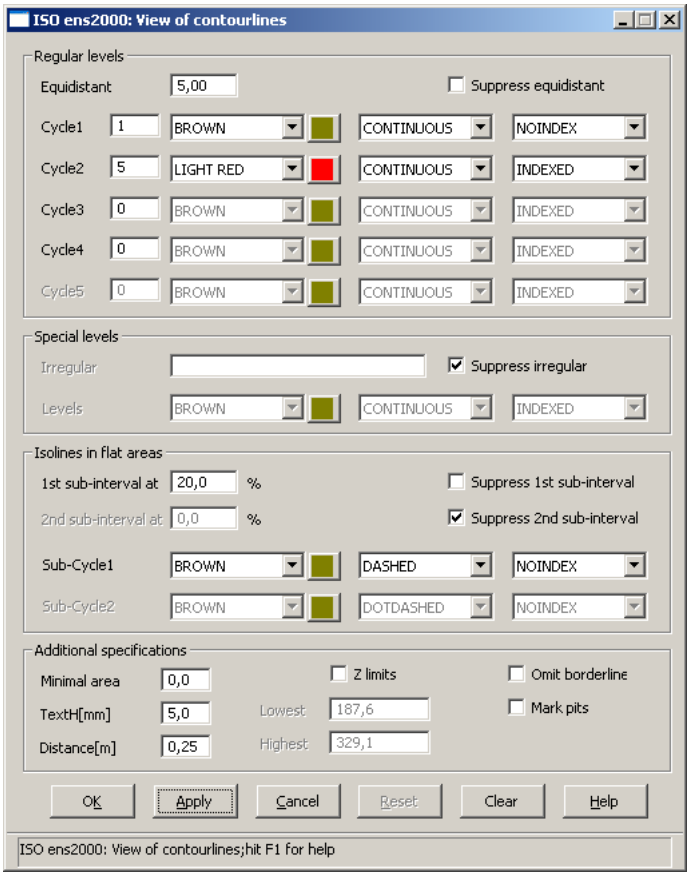


Figure 12.52.: Iso window

Four groups on the window (see figure 12.52) define the basic facilities which are offered by this module:

- The definition and representation of *regular levels*.
- The definition and representation of *special levels*.
- The definition and representation of intermediate *isolines in flat areas*.
- Some additional specifications

12.6.1. Regular Levels

Within the group *Regular intervals* an interval can be specified in the numeric field *Equidistant*. If not limited by Z limits (see section 12.6.4) the contourlines will be computed for

the whole Z range of the selected view limits starting at the first Z value, which can be divided through the interval without remainder and stopping with the last Z value, which fullfills the same restriction.

Equidistant intervals can be turned off by selecting the appropriate checkbox *Suppress equidistant*. It is not allowed to turn off regular levels and spezial levels at the same time.

The next five lines within group *Regular levels* allow to determine the graphic representation for regular isolines. Each line is responsible for a set of isolines. The specification of the relevant set is made in the numeric fields *Cycle1* to numeric fields *Cycle5*. The values in the numeric fields *Cycle1* to numeric fields *Cycle5* define the cycle of the isolines for which the attributes should be applied. The rules for all isolines stem from the logical addition of all specifications. For example a value of 1 in the numeric field *Cycle1* and a value of 5 in the numeric field *Cycle2* state that each fifth line should be displayed with the attributes defined right of the numeric field *Cycle5* and all other with the attributes defined right of the numeric field *Cycle1*.

A value of zero in the numeric fields *Cycle* disables the attributes in the concerning row. The default values for the five Cycle fields are 1 for the numeric field *Cycle1* and 5 for the numeric field *Cycle5*.

The attributes which can be specified for one cycle of isolines are:

Color : BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHT GRAY, DARK GRAY, LIGHT BLUE, LIGHT GREEN, LIGHT CYAN, LIGHT RED, LIGHT MAGENTA, YELLOW, WHITE

Linetype : CONTINUOUS,DASHED,DOTDASHED,DOTTED

Indexstyle : INDEXED, NOINDEX, SHORTINDEX

12.6.2. Special Levels

Additionally to the equidistant intervals there can be defined irregular intervals within the group *Special levels*, which will only be computed if they aren't yet comprised in the equidistant intervals. Mind that for the irregular intervals the Z limits (see section 12.6.4) are also applicated and that only if the irregular intervals don't coincidence with the equidistant contourlines a specific graphic represention can be chosen. The irregular contourlines can be determined by entering their heights in the text field *Irregular*. More than two Z values should be seperated by blanks or tabs. Only valid heights (ie. such that are numbers and lie between the upper and lower bounds of the current terrain model) will be treated.

Irregular intervals can be turned of by selecting the appropriate checkbox *Suppress irregular*. It is not allowed to turn off regular levels and spezial levels at the same time.

The graphic reperesentation of irregular levels can be determined in the same way like with regular levels.

12.6.3. Isolines in Flat Areas

Within the group *Isolines in flat areas* supplemental contours to the equidistant interval can be computed in flat areas of the terrain. Whether such supplemental contours are

computed depends on the slope of the terrain. If the slope (given in of tangens of the slope angle) is less than the value specified with numeric field *1st Sub-Interval* the interval from regular levels is halfed, and if the slope is less than the value specified with numeric field *2nd Sub-Interval* the interval from regular levels is divided by four to get additional contourlines.

The creation of supplemental contours can be suppressed by selecting the appropriate checkbox *Suppress 1st sub-interval* and checkbox *Suppress 2nd sub-interval*.

The graphic reperesentation of supplemental contours can be determined in the same way like with regular levels.

12.6.4. Additional Specifications

The isolines are computed and displayed starting at the level defined in the numeric field *Lowest* up to the level defined in the numeric field *Highest* if the checkbox *Z limits* is selected, otherwise for the whole extension in Z direction of the DTM. The numeric field *Lowest* and the numeric field *Highest* are only accessible if the checkbox *Z limits* is checked. The default values for these numeric fields are the minimal resp. maximal Z values of the terrain model.

With the numeric field *TextH[mm]* the textheight in the map for the project scale (see section 9.2.8) can be defined. The numeric field *Distance* defines the average distance between index labels along the contour in the map sheet given in meters; the default is 0.25 meter.

If the project scale (see section 5.3) is changed and the state-switch *Iso* is in state display, a recalculation of the contourlines will be triggered. If the numeric field *Minimal area* has a vlue greater than 0 then all isolines enclosing less than the value of the numeric field *Minimal area* (area in mm^2 in the plot) are skipped. If the checkbox *Mark pits* is selected, pits are marked with arrows.

Selecting checkbox *Omit borderlines* allows the calculation of isolines regardless of borderlines present in the DTM.

The current specifications are applied, provided that the state-switch *Isolines...* is in the proper state, on pressing the button *OK*. The window is closed without change on pressing the button *Cancel*. With the button *Clear* you can reset the input fields to the default values. Pressing the button *Reset* you can set the input fields to the last accepted (the ones for you pressed the button *OK*) attributes.

Here are some commandfile examples for this window:

cmL example

```
ens2000/,
  isolines/,
  equidistant = (2.5),
  ZLimits = 1,
  Highest = 300.0, Lowest = 200.0,
  Cycle1 = (1), Cycle2= (3), Cycle3= (5),
  Color1= (GREEN), Color2 = (BLUE), Color3 = (RED),
  Linestyle1 = (dotdashed), Linestyle1 = (dashed),
  Index1 = (short),
  textH = (10.0),
  OK;
ens2000/,
  isolines = (d);
```

cmL example

```
@iso_irreg;
 ens2000/
  isolines/
    suppressEqui = 1,
    suppressIrr = 0,
    irregular=(212 227 252 273 288 297),
    ZLimits = 1,
    Highest = 300.0, Lowest = 200.0,
    ColourLev = ("light blue"),
    LinestyleLev = (contin),
    IndexLev=(indexed),
    textH = (2.0),
    OK;
  isolines = (d);
@endProc;
```

cmL example

```
@iso_subinter;
 ens2000/
  isolines/
    Clear,

    // --- specifications for regular levels ------------
    suppressEqui = 0,
    equidistant = 5,
    Cycle1 = (1), Cycle2 = (5),
    Colour1= (brown), Colour2 = ("light red"),
    Linestyle1 = (contin), Linestyle2 = (contin),
    Index1 = (noind), Index2 = (indexed),

    // --- specifications for irregular levels ----------
    suppressIrr = 1,

    // --- specifications for sub intervals -------------
    supp1stSub = 0,
    supp2ndSub = 0,
    1stSubinter = 20,
    2ndSubInter = 10,
    ColourSub1= (white), ColourSub2 = (yellow),
    LinStyleSub1 = (dashed), LinStyleSub2 = (dotdashed),
    IndexSub1 = (noind), IndexSub1 = (noind),

    // --- additional specifications --------------------
    ZLimits = 0,
    textH = (2.0),
    OK;
  isolines = (d);
@endProc;
```

12.6.5. Error Messages

An error which occured during the calculation of contours will be reported via an error message which has the hint to check the corresponding log file (see section 9.2.8). Provided that the protocol option is enabled this will contain a diagnosis of the error. The following error-number, text/comment table is a thorough description of all possible errors:

```
4001    WRONG VALUE FOR PARAMETER INTERVAL
        Contour interval must be positive.

4002    WRONG VALUE FOR PARAMETER GRIDRED
        GRIDRED values must be positive.

4003    WRONG NUMBER OF VALUES IN LIMIT
        Allowed number of values are:
          Without attribute: 4, 6, 8, ..., 20
          With PPDELT: 5, with SIDES: 4.

4004    TOO MANY VALUES IN LEVELS
        A maximum of 40 values is allowed.

4005    GRIDRED VALUE DOES NOT FIT CU
        The ( number of grid lines in CU - 1 ) must be divided by
        the GRIDRED value without remaindering.

4101    INTERNAL ERROR
        A contour line segment within one CU, initiated as closed,
        does not end properly.
        DEM structure not in order.
        Computation continues with a small gap in a contour.

4102    INTERNAL ERROR
        Infinite loop for contour line segment within one grid cell.
        DEM structure not in order.
        Computation continues with a small gap in a contour.

4111    TOO MANY CONTOUR LINE POINTS
        Number of contour line points (IM) for one level (HL)
        exceeds the maximum (IM=1200000).
        Computation stops because internal working storage is too
        small.  Use smaller map sheets or larger grid width.

4112    TOO MANY CONTOUR LINES
        Number of contour line segments (NIT) for one level (HL)
        exceeds the maximum (MIT=800000).
        Computation stops because internal working storage is too
        small.  Use smaller map sheets or larger CU size.

4199    INTERNAL ERROR
        Following a contour line segment initiated as closed within
        a CU, the CU margin is reached.
```

```
        DEM structure not in order.
        Computation continues.

5001    NO PLOT FILE CREATED
        No directive PINIT was given.

5002    PINIT MUST BE SPECIFIED FIRST
        PINIT must be used to initiate map sheet before ISOPLOT or
        SYMBOL is used.

5003    PROGRAM ABANDONED
        Errors in SCOP.DTM, or files not in order.

5004    EMPTY FILE ii
        File number ii is empty. Errors in SCOP.DTM, or files not in
        order.

5010    TEXTH = h  , < 0,5 MM OR > 10,0 MM
        The given text height h is too small or too large.
        In batch mode, the default value is used; interactively, the
        parameter must be repeated.

5011    LEVELS LOWER THAN LOWEST NIVEAU  l  -> IGNORED
        ( list of ignored levels )
        All levels lower than  l  ( given with parameter LOWEST )
        are ignored.

5012    LEVELS HIGHER THAN HIGHEST NIVEAU  h  -> IGNORED
        ( list of ignored levels )
        All levels higher than  h  ( given with parameter HIGHEST )
        are ignored.

5013    NOINTER + NOLEVELS ??  NO ACTION
        Useless directive.

5014    LOWEST LEVEL =  l  >  HIGHEST LEVEL =  h
        Value from HIGHEST < value from LOWEST
        or value from HIGHEST < lowest niveau in map sheet
        or value from LOWEST > highest niveau in map sheet.

5015    LEVELS  h  NOT FOUND
        Niveau  h  given by LEVELS was not computed in CONTOURS.

5020    EXISTING PLOT FILE CLOSED
        RETURN without NOCLOSE, STOP or a new PINIT automatically
        closes an already existing plot file.

5050    WRONG VALUES OF SPOT/SINGUL
        Number of digits after decimal point < -1 or > 3
        or number of digits before decimal point negative.

5101    M=m, HL=h, XA=x1, YA=y1, XE=x2, YE=y2
        Internal error: A contour initiated at x1,y1 as open stops
```

uncorrectly at x2,y2. This might happen with supplemental contours; for other contours, it shows internal errors in the DEM.

5104 HL=h, L=l, ML=ml
In the niveau h, the number of digits l before the decimal point is too large.
Only six digits before the decimal point are plotted.

5108 INTERNAL ERROR
Contour initiated as closed does not end properly.
See also 5101.

5109 M=m, HL=h, XA=x1, YA=y1, XE=x2, YE=y2
Internal error. Closed contour does not end properly (distance first to last point too large). See also 5101.

5110 ERROR IN ATAN2
Internal error. One contour index may be plotted in wrong direction.

5111 TOO MANY ARROWS
A maximum of 16650 (8325 on some small computers) may be plotted; the others are omitted.

12.7. Hill Shaded Views

Hill shaded views are gray-scale digital images in which each pixel has a gray level according to the exposition and steepness of the terrain. For producing a hill shaded view, the position of the light source can be defined.

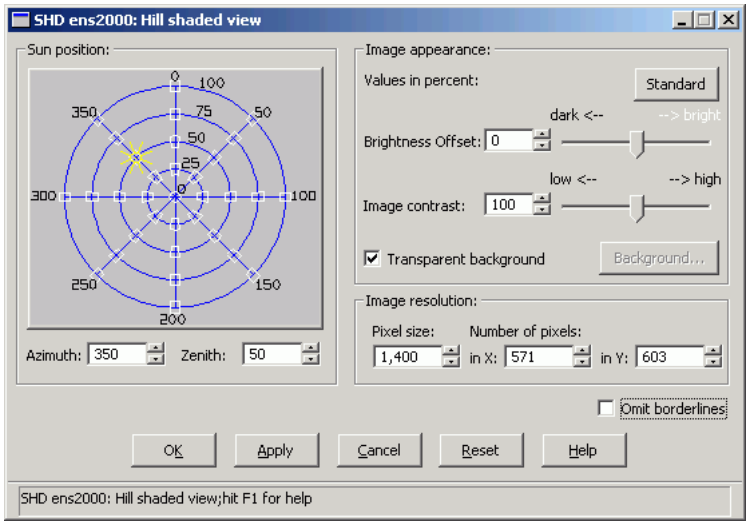


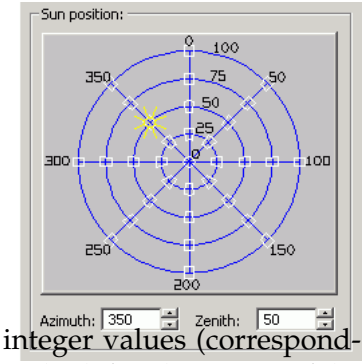
Figure 12.53.: Shading window.

The window *SHD overlayName: Hill shaded view* for defining both the light source, brightness, contrast, etc. (cf. fig. 12.53) can be opened behind the state-switch *Shade...* of a model overlay. In the left part of the window , the position of the light source can be defined (cf. section 12.7.1). The appearance of the image (its brightness, contrast and background gray tone) can be adjusted (cf. section 12.7.2). In the group *Image resolution* (cf. section 12.7.3), the extent of a pixel or the number of pixels can be selected (only in mode *final*, cf. section 4.5).

By clicking the button *OK* or the button *Apply*, the parameters are accepted and – if the state-switch *Shade...* has state *display* – a new hill shaded view will be derived and displayed on the main graphics panel. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones. The button *OK* and the button *Cancel* will additionally close the window .

In order to export a hill shaded view to a specified file, refer to section 12.2.7. There, the file name and file type can be selected. Additionally, the view can be added as image overlay (cf. section 13).

12.7.1. The group *Sun Position*



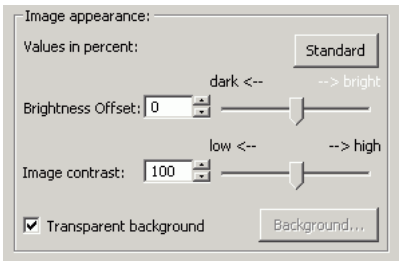
The position of the light source is defined by

- its azimuth which can be selected in the numeric field *Azimuth* in gon (between 0 and 400) with 0 gon at north, 100 gon at east, ...
- and its zenith angle which can be selected in the numeric field *Zenith* in gon (between 0 and 100) with 0 gon in the zenith and 100 gon at the horizon.

Both values can be selected by clicking in the graphical sketch above. By clicking inside the white boxes, rounded integer values (corresponding to the intersection of the blue lines) are selected. The currently defined position of the light source will always be drawn as yellow sun symbol.

Usually a light source in the north west will be selected, as humans are used to these shaded maps. The higher the zenith angle (i. e. the closer the sun at the horizon), the more contrast will be visible.

12.7.2. The group *Image Appearance*

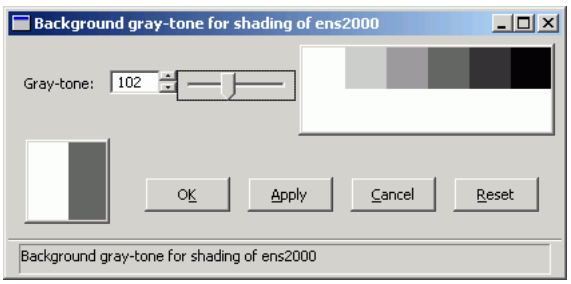


Brightness: By pulling the slider *Brightness offset*, the image can be made darker or brighter. If the slider is in the middle (offset 0 %), the medium gray-tone will be set to 128, the minimum to 1 and the maximum to 255. If it is e. g. at the left end (offset -100 %), all values from the minimum to the medium gray-tone will be set to 0, and the maximum to 128 (these examples suppose image contrast 100 %).

Contrast: By pulling the slider *Image contrast*, the contrast can be modified in a range between 0 % (no contrast) and 200 % (double contrast than standard). The more left, the lower the contrast.

Standard values: Pressing the button *Standard* will reset the brightness offset to 0 % and the image contrast to 100 %.

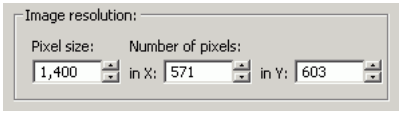
Background: Hill shaded views are stored as rectangular digital images. If the DTM limits are not rectangular, those pixels of the hill shaded image which are not part of the DTM are set to zero. Usually (if the checkbox *Transparent background* is on), the background pixels (i. e., the pixels of "color number" 0) are not drawn. If the checkbox *Transparent background* is switched off, the button *Background...* will be made accessible. In the window *Background gray-tone for shading of overlayName* behind it, any gray level for these background pixels can be selected.



Instead of changing the numeric field *Gray-tone*, the respective slider can be pulled. In the right graphics window , 6 predefined gray tones are drawn. In the second line, the last selected gray tones are drawn (up to 6). Any of these gray tones can directly be selected by clicking at it with the left mouse button. In the lower graphics window , the previously accepted gray tone is drawn in the left half, and the current gray tone is drawn in the right half.

The button *OK* and the button *Apply* accepts the current gray tone, whereas the button *Cancel* and the button *Reset* will omit changes of the gray tone. The window is closed either by button *OK* or by button *Cancel*.

12.7.3. The group *Image Resolution*



The size of the pixels and the extent of the image (i. e., the number of pixels) are related to each other. The numeric fields *Pixel Size, Number of pixels in X, in Y* are always updated automatically.

In mode *screen* (cf. section 4.5), these numeric fields are locked, because the resolution of the image is selected by the system for providing a fast display on the screen. Only in mode *final*, the user can select the resolution. Nevertheless, as long as the user has not changed the values proposed by the system, the system will propose new values on each change of the extent of the view (e. g. after changing the view limits). On the other hand, once the user has changed one of these values, he is responsible himself for updating these values.

NOTE: If the user wants to select the extent of the image by cmL/cmF (cf. section 7), the label *Number of pixels* of the numeric fields *in X, in Y* must not be specified.

12.7.4. The checkbox *Omit borderlines*

When checkbox *Omit borderlines* is selected the hill shaded view of the DTM is calculated regardless of borderlines present in the DTM.

Finally, a cmL/cmF example of the window *SHD overlayName: Hill shaded view* is given.

cmL example

```
ens2000/
  Shade/
    Azimuth(350),
    Zenith(50),
    BrightnessOffset(0),
    ImageContrast(100),
    OK;
  Shade,d;
```

12.8. Elevation Coded Views (Z-coding)

Elevation coded views are maps in which different elevation levels are drawn in different colors. Typical color tables reach from saturated green for low elevations to brown colors for mountains. The color coded maps are represented as digital images. Each pixel is assigned a color derived from the elevation of the corresponding position in the DTM. Neighboring pixels will have the same color as long as the corresponding elevation lies in the same elevation interval. In SCOP++, the z-levels which are merged to one elevation class can be selected either automatically by different strategies, or may also be defined irregularly. Similarly, the color table assigned to the table of elevation classes may be a predefined one, or the colors may be edited.

Z-coded views can be created from the main ("primary") model (e. g. an elevation model). Several secondary models (e. g. slope model, quality models, etc.) offer only a simple visualization possibility which is a subset of the Z-coding window. This manual includes only one description (describing all functionalities) even if there are no all elements available on some specific parameter windows.

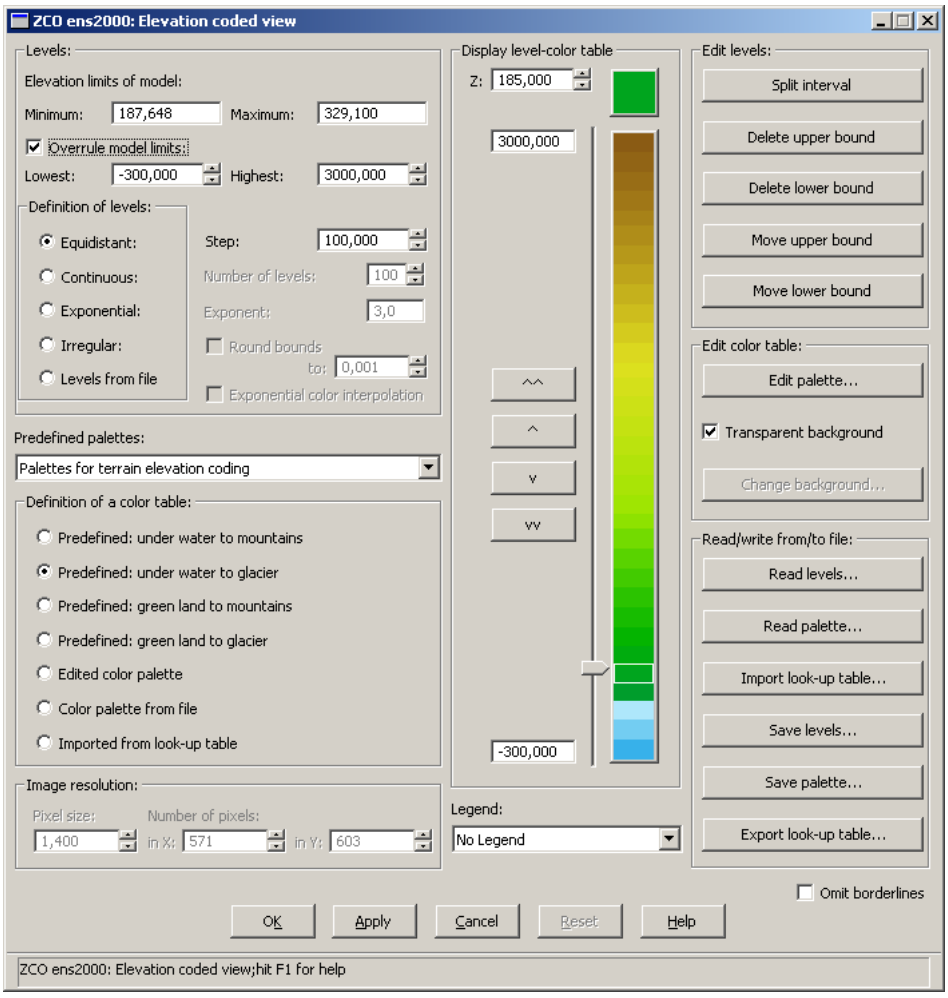


Figure 12.54.: Z-coding window

The window *ZCO overlayName: Elevation coded view* for defining parameters of color coded maps can be opened behind the state-switch *Z-code...* of a model overlay. (cf.fig. 12.54). Two main parameter sets have to be defined: one the one hand elevation classes and on the other hand a color palette. Theoretical backgrounds are explained in the next two sections: the way how color palettes are declared in SCOP++ is described in section 12.8.1. In section 12.8.1 it is explained how such a palette is merged with the elevation classes.

In the following sections, the practical usage of the user interface is explained in detail: How to define the elevation classes (group *Levels*, cf. section 12.8.3), how to edit them (group *Edit levels*, cf. section 12.8.4), how to select a palette (group *Definition of a color table*, cf. section 12.8.5) and how to adapt it (group *Edit color table*, cf. section 12.8.6). In the group *Display level-color table* (cf. section 12.8.2), the current "legend" can be seen, i. e. the color assigned to each elevation is displayed. In the group *Image resolution* (cf. section 12.8.7), the extent of a pixel or the number of pixels can be selected (only in mode *final*, cf. section 4.5). In the group *Read/write from/to file* (cf. section 12.8.8), the defined elevation classes, the interpolating color palette and the resulting look-up table can be imported from or exported to file.

By clicking the button *OK* or the button *Apply*, the parameters are accepted and – if the state-switch *Z-code...* has state *display* – a new z-coded view will be derived and displayed on the main graphics panel. Pressing the button *Cancel* or the button *Reset* will reset the parameters to the last accepted ones. The button *OK* and the button *Cancel* will additionally close the window .

For most applications, regular elevation classes and one of the predefined color tables can be used. An example of how to define irregular elevation classes and an own color table will be given in section 12.8.10.

In order to export a z-coded view to a specified file, refer to section 12.2.7. There, the file name and file type can be selected. Additionally, the view can be added as image overlay (cf. section 13).

12.8.1. Interpolating Color Palette vs. Look-up Table

For defining an elevation coded view, two independent sets of parameters have to be specified. On the one hand, the elevation range is mapped into elevation classes ("levels"). On the other hand, a (usually smooth) color palette is defined. By merging these two, each of the elevation classes is assigned its own color. For each pixel, the index of the elevation class it belongs to is stored. Additionally, a so-called look-up table is stored providing for each elevation class index the respective color.

A look-up table has as many entries as there are elevation classes. Thus, changing the number of elevation classes forces to change the look-up table, too.

In order to separate the color definition from the elevation class definition, in SCOP++ a so-called "interpolating color palette" (ICP) is used. Such a palette is independent of elevation classes and can easily be transferred between different SCOP++ projects.

The ICP has a simple ASCII interface as explained in section section 12.8.1. The look-up table written on the image data file is finally created by intersecting the palette with the elevation classes, which is described in section 12.8.1.

The Syntax of an Interpolating Color Palette (ICP)

The ICP consists of a series of color nodes. Two types of color nodes are available differing in the way the corresponding elevation is defined:

- Intermediate Nodes: The elevation of these nodes is given as percentage in the elevation interval
- Fixed Nodes: The elevation of these nodes is fixed to a specified height value.

Intermediate Nodes: The simplest ICP consists of two intermediate nodes, one at elevation 0 %, the second at elevation 100 %, e. g. an ICP "from black to white":

```
AT 100% R/G/B [0 - 255] 255/255/255 // white
AT   0% R/G/B [0 - 255]   0/  0/  0 // black
```

In this example, when intersecting with elevation classes, the lowest elevation will be assigned black, the highest elevation white. An elevation of 10 % would be assigned a light gray (26/26/26).

Some more words to the syntax: The color of an intermediate node is given by three values (separated by a single slash) either in red/green/blue (three values in the range between 0 and 255) or in intensity/hue/saturation: intensity and saturation in percent (between 0 and 100), hue in degrees on the color wheel (0 or 360 for blue, 120 for green, 240 for red). The color space used is defined by the key word `R/G/B` or `I/H/S`, respectively. The units can optionally be given between brackets (`[` and `]`). All text after the double slash (`//`) is interpreted as comment.

Two intermediate nodes at 0 % and 100 % must always be contained in the series of color nodes. Additionally, further intermediate nodes can be given for color interpolation. For instance, the standard ICP "green land to mountains" is given by:

```
AT 100% I/H/S [%/deg/%]  30/205/ 75 // dark brown
AT  60% I/H/S [%/deg/%]  50/180/ 75 // dirty yellow
AT  33% I/H/S [%/deg/%]  45/160/100 // light green
AT   0% I/H/S [%/deg/%]  30/100/100 // dark green
```

Fixed Nodes: These nodes are fixed concerning their elevation. Additionally, they allow jumps in color interpolation. Each fixed node splits the interpolation range into two interpolation ranges – one from the lowest elevation (0 %) to the elevation of the fixed node, the other from that fixed elevation to the highest elevation (100 %). With the fixed node two new intermediate nodes have to be (with 100 % for the lower interpolation range and 0 % for the higher one). Thus, the syntax of a fixed node consists of three lines, which can be seen in the following example (the standard ICP "under water to mountains"):

```
AT 100% I/H/S [%/deg/%]  30/205/ 75 // dark brown
AT  60% I/H/S [%/deg/%]  50/180/ 75 // dirty yellow
AT  33% I/H/S [%/deg/%]  45/160/100 // light green
AT   0% I/H/S [%/deg/%]  30/100/100 // dark green
FIXED NODE AT 0                     // sea level
AT 100% I/H/S [%/deg/%]  90/ 45/100 // light blue
AT   0% I/H/S [%/deg/%]  50/ 40/ 80 // dark blue
```

Merging an ICP with Elevation Classes to a Look-up Table

For intersecting an ICP with levels definition (elevation classes), the following rules are applied:

1. Get the lowest and highest elevation from the levels definition.
2. If there is no fixed node lower than the lowest elevation, set the lowest 0%-intermediate node to the lowest elevation.
3. If there is no fixed node higher than the highest elevation, set the highest 100%-intermediate node to the highest elevation.
4. For each elevation class: Calculate the medium elevation (i. e. the average between the higher and the lower interval bound) and interpolate the color.

The result will be a look-up table containing the background color at entry 0 and starting at entry 1 for each elevation class the interpolated color. Although the color of intermediate nodes can be given either in the RGB or in the IHS space, color interpolation will always be performed in the IHS space (double-cone IHS model). For interpolating the hue, the shorter path on the color wheel is used (e.g. half way between the hues 340 and 40 = 10).

If the color for an elevation equal to the elevation of a fixed node is requested, the color of the higher interpolation interval is returned (i.e. the color of the respective 0%-intermediate node).

Special Rules for Exponential Levels and Exponential Look-up Table Creation

The color interpolation in the palette can also be performed exponentially (e. g. if the levels definition was declared exponential). In that case the last of the above rules for merging levels with ICP is not true, because the ICP is interpreted also exponentially. In that case the following rule is applied:

4. For all elevation classes: Invert the formula for the calculation of exponential level bounds (cf. section 12.8.3) in order to create a fictitious "linear" levels definition. Calculate medium elevation values for each of these fictitious intervals and interpolate for these fictitious mediums the color.

Examples: For all the following examples, the same ICP (under water to glacier) is used:

```
AT 100% R/G/B [0 - 255] 255/255/255 // white
AT   0% R/G/B [0 - 255] 255/255/255 // white
FIXED NODE AT 3000                  // glacier level
AT 100% I/H/S [%/deg/%]  30/205/ 75 // dark brown
AT  60% I/H/S [%/deg/%]  50/180/ 75 // dirty yellow
AT  33% I/H/S [%/deg/%]  45/160/100 // light green
AT   0% I/H/S [%/deg/%]  30/100/100 // dark green
FIXED NODE AT 0                     // sea level
AT 100% I/H/S [%/deg/%]  90/ 45/100 // light blue
AT   0% I/H/S [%/deg/%]  50/ 40/ 80 // dark blue
```

(a) Elevation classes: level bounds at 0, 500, 1000 and 1500

Resulting look-up table with 4 entries:
entry 0: background color
entry 1: color for elevation 250: I/H/S 34 % / 115 deg / 100 %
entry 2: color for elevation 750: I/H/S 41 % / 145 deg / 100 %
entry 3: color for elevation 1250: I/H/S 47 % / 167 deg / 91 %
All three colors are green.

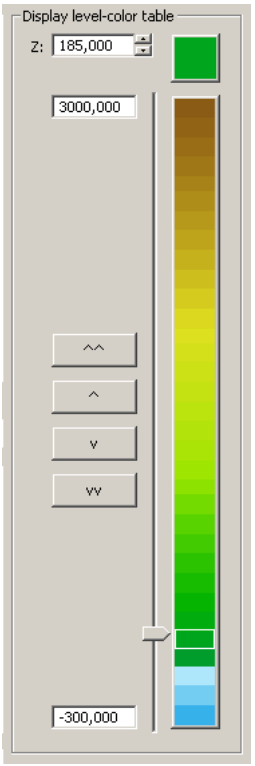
(b) Elevation classes: level bounds at -500, 0, 500, 1000 and 1500

Resulting look-up table with 5 entries:
entry 0: background color
entry 1: color for elevation -250: I/H/S 70 % / 43 deg / 83 %
entries 2 - 4: same as entries 1 - 3 in example (a)
The color of entry 1 will be a light blue.

(c) Elevation classes: level bounds at -500, 0, 500, 1000, 1500, 2000, 2500, 3000 and 3500

Resulting look-up table with 9 entries:
entry 0: background color
entry 1 - 4: same as in example (b)
entry 5: color for elevation 1750: I/H/S 50 % / 178 deg / 76 % (yellow)
entry 6: color for elevation 2250: I/H/S 43 % / 190 deg / 75 % (light brown)
entry 7: color for elevation 2750: I/H/S 34 % / 200 deg / 75 % (dark brown)
entry 8: color for elevation 3250: R/G/B 0/0/0 (white)
Only in this example, the whole palette from blue to white is used.

12.8.2. The group *Display Level-Color Table*



This group has two main purposes:

- to show the currently selected elevation classes (levels) and the colors assigned to them,
- to select an elevation value for editing the elevation classes (cf. section 12.8.4),

The currently selected elevation value is displayed in the numeric field *Z* on top of the group *Display level-color table*. The small color field right of the numeric field *Z* shows the color of the current elevation interval.
The elements below show the whole range of available elevations: the numeric fields display the minimum and maximum elevation of the selected elevation classes and are not editable. The high rectangular graphics window serves as legend showing the colors for all elevation classes. A rectangular window is drawn (in white or black) to indicate the elevation class of the currently selected elevation. The slider left of it is coupled to the numeric field *Z* and can also be used to select a new elevation.
The four buttons left of the slider allow to select an interval bound as new elevation: button ^^ selects the highest interval bound (i. e. the maximum elevation), button ^ selects the next higher interval bound, button v the next lower bound and button vv selects the lowest interval bound (i. e. the minimum elevation). These four buttons are useful especially for irregularly defined elevation classes in order to check all bounds.

Below of the group *Display level-color table* there is the selection *Legend:* which allow to choose if a legend should be generated and where to put it.

12.8.3. The group *Levels*

The total elevation range of the model must be divided into several classes. Up to 255 intervals can be used. The minimum and maximum elevation of the Z-coded map to be produced can be equal to the total elevation range of the model (which is displayed in the non-editable numeric fields *Minimum, Maximum*). If the checkbox *Overrule model limits* is selected, other values can be used for the lowest and highest value in the Z-coded map (in the numeric field *Lowest* and the numeric field *Highest*, respectively). Once the user has entered his own values in the numeric fields *Lowest, Highest*, these values are not updated any more if a new model (with new elevation limits) has been interpolated. The elevation range defined by either the model limits or by the user is used for deriving the look-up table (cf. section 12.8.1). If there are Z-values in the DTM, which are outside the elevation range, the corresponding pixels are set to background color.

Further parameters of the group *Levels* define the strategy of determining the elevation classes. Four strategies can be selected:

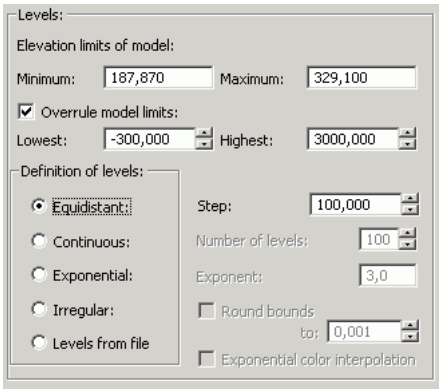


Figure 12.55.: Elevation class definition.

- *Equidistant* classes defined by a constant elevation step to be given in the numeric field *Step*. The number of intervals is derived automatically dependent on the elevation step. All interval bounds are rounded to values which can be divided by the elevation step.
- *Continuous* classes defined by the number of intervals which can be chosen in the numeric field *Intervals*. The elevation range between the given minimum and maximum elevation is divided by the selected number of intervals so that all intervals will have the same width. The elevation step and thus also the interval bounds will usually be real numbers.
- *Exponential* classes defined by the number of intervals N to be chosen in the numeric field *Intervals* and the exponent x to be chosen in the numeric field *Exponent*. The interval bounds are defined by the following exponential function:

  $z_i = z_{min} + (\frac{i}{N})^x (z_{max} - z_{min})$

  with z_{min} and z_{max} the minimum and maximum elevation. If x is chosen greater than 1, there will be more and narrower intervals towards z_{min}, and fewer, wider intervals towards z_{max}. The opposite is true for values of x between 0 and 1.

  Note that the number of decimals used in the Z-coding window (which is defined automatically depending on the model's Z-range) may be too small to separate all intervals. For large exponents, some of the lowest intervals may have a zero width (regarding the displayed number of decimals). Nevertheless, these intervals are valid.

  If exponential classes are selected, usually exponential look-up table creation will be used (cf. 12.8.1). If, however, the standard (linear) merging rules are to be used, the checkbox *Exponential color interpolation* has to be de-selected.
- *Irregular* classes as defined by the operator. In order to define these classes, the operator has to use the tools from the group *Edit levels* (cf. section 12.8.4). Furthermore, the operator can declare the edited interval bounds to be rounded. The value given in the numeric field *to* defines the number of significant digits (e. g. *Round bounds to:* 0.01 will round the value 147.28841 to 147.29).

  Any case, all values will be rounded regarding at least the number of displayed decimals (which is defined automatically depending on the model's Z-range). Thus, in-

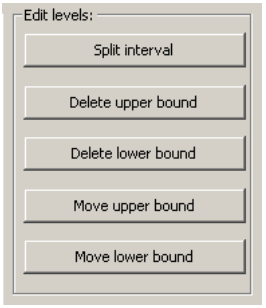
tervals with "zero" width (as they may be have been created by exponential levels definition) are removed.

For irregular classes, standard or exponential color interpolation can be selected (cf. the previous paragraph about exponential classes).

Note that selecting one of the buttons of the group *Edit levels* will automatically change the selected strategy to *Irregular*. See the example in section 12.8.10 on how to edit elevation classes.

- *Levels from file*: The interval bounds may be read from file. Selecting this strategy will open the file browser *Read levels...* from group *Read/write from/to file* for choosing a levels file (cf. section 12.8.8 for details on the file format). On the other hand, this strategy will be selected automatically, when the user chooses a levels file via the file browser *Read levels...*.

12.8.4. The group *Edit Levels*



The tools of the group *Edit levels* are designed to modify existing interval bounds and to define irregular elevation classes.[5] Performing one of the actions described below will automatically change the strategy of levels definition to *irregular* (cf. section 12.8.3).

Before selecting one of the following actions, the current elevation in the group *Display level-color table* (cf. section 12.8.2) has to be set. All actions will be applied to the elevation class which the current elevation belongs to.

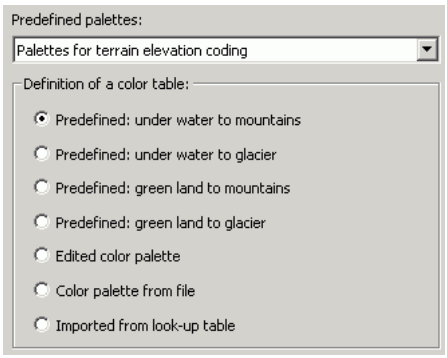
Five actions are possible:

- *Splitting an interval* into two intervals: By pressing the button *Split interval*, the current elevation class is split at the elevation as selected in the group *Display level-color table*. An error message will be sent, if the current elevation already coincides with an interval bound.
- *Merging intervals:*
  - Merging the current elevation class with the next higher one by pressing the button *Delete upper bound*.
  - Merging the current elevation class with the next lower one by pressing the button *Delete lower bound*.
- *Moving interval bounds:*
  - Moving the upper bound of the current elevation class to the elevation as selected in the group *Display level-color table* by pressing the button *Move upper bound*.
  - Moving the lower bound of the current elevation class to the elevation as selected in the group *Display level-color table* by pressing the button *Move lower bound*.

[5] These tools are only available if the group *Levels* contains the selection element *Irregular*.

An error message will be sent, if the current elevation coincides with an interval bound.

12.8.5. The group *Definition of a Color Table*



Besides determining the levels (i. e. the elevation intervals), a color has to be assigned to each of these classes. The colors are defined independently from the levels definition by selecting a so-called interpolating color palette (ICP, cf. section 12.8.1). For deriving the look-up table which is finally written on the resulting image data file, the levels definition and the ICP are merged (cf. section 12.8.1).
For many applications, one of the predefined color palettes (ICP) will be appropriate. Nevertheless, for specific applications, the user wants to define a specific ICP, or he wants to load an ICP from file.
Different groups of ICPs are offered depending on the type of the model. For the Z-coding window of the main ("primary") model, the group of offered ICPs can be selected in the selection *Predefined palettes*.

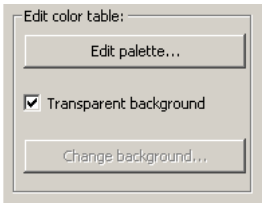
Following strategies of defining an ICP are available:

- *Predefined: xxx*: selecting one of the predefined gray or color ICPs.
- *Edited color palette*: by modifying the currently selected ICP a specifically adapted palette can be created. The modifications can be done after opening the editing window *Palette of z-coding in overlayName* behind the button *Edit palette...* (cf. section 12.8.6).

  Note, that this strategy will be selected automatically, if the user edits the ICP in the window *Palette of z-coding in overlayName*.
- *Color palette from file*: reading an ICP from file. The file could have been created by saving the ICP in a previous project or an ICP is prepared in a text editor by applying the simple ASCII syntax of an ICP, cf. section 12.8.1).
- *Imported from look-up table*: If a look-up table is available, it can be imported and converted to an ICP. This is described in section 12.8.8.

12.8.6. The group *Edit Color Table*

Both the color palette and the background color may be edited.



The window *Palette of z-coding in overlayName*: [6] Pressing the button *Edit palette...* with either the left or the right mouse button opens this window for adapting the "interpolating color palette" (ICP).

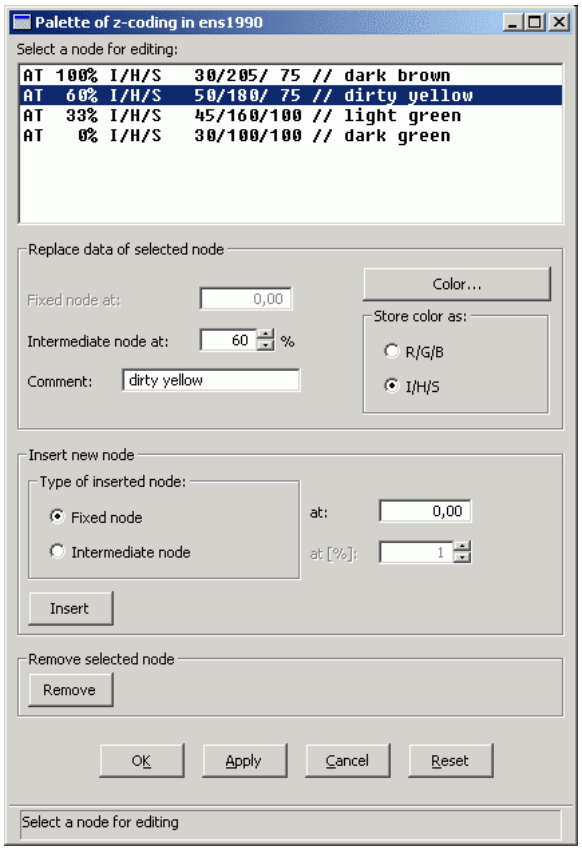


Figure 12.56.: Editing the color palette.

At the top of the window , the current ICP is written in its ASCII syntax (cf. section 12.8.1). In this box, exactly one node can be selected for further editing. Below, various elements for editing the ICP are provided. Existing nodes can be changed, further nodes can be inserted, or nodes can be removed:

- The group *Replace data of selected node*: Depending on whether a intermediate node or a fixed node is selected different elements are made available:
  - the numeric field *Fixed node at*: The elevation of a fixed node can be edited here.

[6] Editing the palette is only available if the group *Definition of a color table* contains the selection element *Edited color palette*.

  - the numeric field *Intermediate node at*: the relative position of intermediate nodes can be edited here.
  - the text field *Comment*: for each node, a comment can optionally be given. It is written at the end of the line after a double slash (//).
  - the button *Color...*: pressing the button with either the left or the right mouse button opens the window *Color of node* for selecting a color either in RGB or in IHS (cf. the last paragraph of this section for reading how to define a color).
  - the selection *Store color as* in order to define whether the color is written as red/green/blue values or as intensity/hue/ saturation values. Changing this selection does not change the color of the node, only its representation in the ASCII syntax.

- The group *Insert new node*: After selecting the *Type of inserted node*, the elevation can be defined in the numeric field *at* for a fixed node or in the numeric field *at [%]* for an intermediate node. By pressing the button *Insert* with the left mouse button, the node is inserted at the specified position. When inserting a fixed node, two new intermediate nodes are inserted together with it, because at a fixed node, a color jump can be introduced (cf. section 12.8.1). Furthermore, the relative positions of intermediate nodes below and above it are changed, so that they will remain on the same elevation.

  The color of the new node is derived by interpolating the color at the respective position. Thus, inserting a new node does not change the palette. After inserting the node, the operator has to change its color in order to change the palette.
- The group *Remove selected node*: After selecting a node at the top of the window , the button *Remove* can be pressed with the left mouse button in order to delete that node from the ICP. If a fixed node is removed, the two intermediate nodes belonging to it will remain in the ICP, but their relative positions are changed, so that they will remain on the same absolute elevation.

After accepting the edited ICP (by selecting the button *OK* or the button *Apply*), it is merged with the current levels (elevation intervals) and the result can be seen in the legend (i. e. in the group *Display level-color table*, cf. section 12.8.2). For starting deriving a new z-coded map, the button *OK* or button *Apply*) at the main parameter window *ZCO overlayName: Elevation coded view* has to be selected, too. The button *Cancel* and the button *Reset* will reject changes of the ICP. The window is closed either by button *OK* or by button *Cancel*.

Note that editing the ICP by command line commands (cf. section 7) is quite complicated. For preparing a command file procedure with a specifically adapted ICP, we recommend to prepare this ICP in a text editor, write it to a file and read this ICP-file.

Background color: Z-coded views are stored as rectangular digital images. If the DTM limits are not rectangular, those pixels of the Z-coded image which are not part of the DTM are set to zero. Usually (if the checkbox *Transparent background* is on), the background pixels (i. e., the pixels of "color number" 0) are not drawn. If the checkbox *Transparent background* is switched off, the button *Change background...* will be made accessible. In the window *Background color of z-coding of overlayName* behind it, any color for these background pixels can be selected. See the next paragraph on how to select a color.

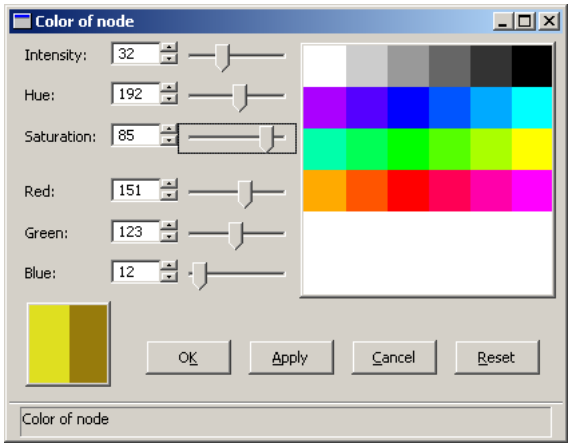


Figure 12.57.: Selection of a new color.

The window *Color of node*: A color can be adjusted both in RGB (red-green-blue) color space or in IHS (intensity-hue-saturation) color space. By changing one of the values in the numeric fields *Red, Green, Blue* the values in the numeric fields *Intensity, Hue, Saturation* are updated automatically. Analogously, changing one of the values in the numeric fields *Intensity, Hue, Saturation* will change all three values in the numeric fields *Red, Green, Blue*.

Instead of changing the numeric field , the respective slider can be dragged.

In the right graphics window , 6 gray values and 18 predefined colors are drawn in the first four lines. In the fifth and sixth line, the last selected colors are drawn (up to 12). Any of these colors can directly be selected by clicking at it with the left mouse button.

In the lower graphics window , the previously accepted color is drawn in the left half, and the currently adjusted color is drawn in the right half.

The button *OK* and the button *Apply* accepts the current color, whereas the button *Cancel* and the button *Reset* will reject changes of the color. The window is closed either by button *OK* or by button *Cancel*.

12.8.7. The group *Image Resolution*

The elements of the group *Image resolution* and their behavior are the same as in the window *SHD overlayName: Hill shaded view* (cf. section 12.7.3).

12.8.8. The group *Read/Write from/to File*

All parameters – the levels definition (*levels*), the color palette (*palette*) and the result of merging these two, the look-up table – can be exported to file in order to be read again in another project or by a command file procedure. After clicking at one of the six buttons *Read levels..., Read palette..., Import look-up table..., Save levels..., Save palette..., Export look-up table...*[7] with the left mouse button, a file name can be selected. All files are ASCII, the file formats are defined

- for the levels (`*.lev`): In the first line of the file, key words are written. In the following lines, the level bounds are written starting with the lowest interval.
  Example of two intervals (the first from 100 to 150, the second from 150 to 200):
  ```
  SCOP++ ZCO intervals
  100.000
  150.000
  200.000
  ```
- for the interpolating color palette (`*.pal`): A file containing the ICP in its ASCII syntax (cf. section 12.8.1).
- for the look-up table (`*.lut`): In each line, a color number and three color values (red, green and blue) in the range between 0 and 255 must be given. The file must start with color number 0 which will be used as background color. Afterwards, up to 255 colors (color numbers: 1, 2, ... 255) may be defined. If the current color table is written to file, exactly as many colors are written as there are elevation intervals.
  Example with white background, first interval in red, second interval in blue:
  ```
  0  255  255  255
  1  255    0    0
  2    0  255    0
  ```

If the file selected via the button *Read levels...*, the button *Read palette...* or the button *Import look-up table...* is not valid, an error message will be sent. Otherwise, the file will be interpreted and the current levels/color table definition will be changed (cf. sections 12.8.3 and 12.8.5).

The windows behind each of these six buttons includes a text field and the file browser *Browse...* for selecting the file name and the buttons *OK/Cancel* for accepting/rejecting the file name.

For importing a look-up table, one of two import strategies can be additionally selected in the window *Import look-up table*. Strategy (a) tries to convert the look-up table to a general palette, whereas strategy (b) maps each color of the look-up table directly to the current elevation intervals.

[7] The buttons are only available if the group *Levels* contains the selection element *Levels from file* and the group *Definition of a color table* contains the elements *Color palette from file* and *Imported from look-up table*.

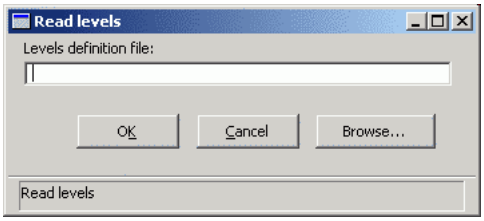


Figure 12.58.: Selecting a filename (e. g. for reading levels)

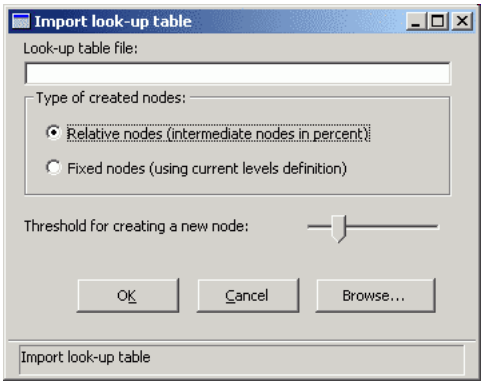


Figure 12.59.: Importing a look-up table

(a) Conversion to intermediate nodes: The look-up table is converted to an interpolating color palette by inserting a intermediate node each time the color at entry i is different from the interpolation result between the colors at entries i - 1 and i + 1. A tolerance threshold for colors to be accepted as equal can be given by the slider at the bottom of the window . The more left the slider , i. e. the smaller the tolerance, the more intermediate nodes will be inserted. Contrarily, the more right the slider , i. e. the higher the tolerance, the fewer nodes will be inserted, but some of the colors of the imported look-up table might be lost.

(b) Conversion to fixed nodes: This import strategy is completely different to the first one. It uses the current levels (elevation intervals) and introduces at each level bound a fixed node. Thus, each elevation interval corresponds to one color interpolation interval. For each color interval the color is set constant. The lowest interval gets the color of look-up table entry 1 (entry 0 is used for the background color), the next interval gets the color of entry 2, etc. If the look-up table has not enough entries, the highest intervals all get the same color. If the look-up table has more entries than there are elevation classes, not all colors will be used.

12.8.9. The checkbox *Omit borderlines*

When checkbox *Omit borderlines* is selected the elevation coded view of the DTM is calculated regardless of borderlines present in the DTM.

12.8.10. Example

In this section, an example of defining irregular elevation classes and assigning specific colors to these classes is given. The example is shown on a DTM with the lowest elevation 187.870 m and the highest point at 329.100 m.

Defining irregular elevation classes: Suppose, we want to define the following elevation classes: The elevation range from 200 m to 300 m should be represented in intervals of step 10 m, all points lower than 200 m and all points higher than 300 m should be merged to one class, respectively. Additionally, the elevation range between 240 m and 260 m should be represented by steps of 4 m.

This example is best started by selecting equidistant elevation classes with step 10 m (cf. section 12.8.3). Afterwards, in the group *Display level-color table* (cf. section 12.8.2), we select the current elevation at ca. 195 m in order to be in the interval below 200 m. We press the button *Delete lower bound* (cf. section 12.8.4) for merging the current interval (from 190 m to 200 m) with the next lower one. The levels definition strategy is automatically changed from *Equidistant* to *Irregular*. Similarly, we select ca. 305 m as new current elevation and press the button *Delete upper bound* twice for merging the highest intervals.

Finally, the range between 240 m and 260 m has to be modified. The two intervals (from 240 m to 250 m and from 250 m to 260 m) have to be split into five intervals of 4 m width each. We select the current elevation at 244 m (exactly) and press the button *Split interval*. The same at elevation 248 m and at 256 m. In order to move the level bound from 250 m to 252 m, we select the current elevation at 252 m and select button *Move lower bound*.

cmL example

```
@irregularIntervals;
ens2000/
  Z-code/
    DefinitionOfLevels(Equidistant),Step(10),
    Z:(195),DeleteLowerBound,
    Z:(305),DeleteUpperBound,DeleteUpperBound,
    Z:(244),SplitInterval,
    Z:(248),SplitInterval,
    Z:(256),SplitInterval,
    Z:(252),MoveLowerBound;
@endProc;
```

Defining a special color table: Now we want to assign a specific color table to the irregular elevation classes defined above. All elevations below 200 m and all above 300 m shall appear black. From 200 m to 240 m and from 300 m down to 260 m the colors shall be interpolated from red to yellow. The five intervals between 240 m and 260 m shall appear from bright to dark green.

We start by assigning a predefined color palette with only two colors, e. g. *Predefined: black-white* (cf. section 12.8.5) and immediately change to *Edited color palette*. The black-white color palette is the basis for creating an own interpolating color palette (ICP). For

editing the palette, the window *Palette of z-coding in overlayName* is opened (cf. section 12.8.6).

First, we change the color of the highest node to black. We select the highest node (100%), open the window *Color of node* behind the button *Color...*, select a black color and press the button *Apply*. Furthermore we change the comment of the node from "white" to "black":

```
AT 100% R/G/B   0/  0/  0 // black
AT   0% R/G/B   0/  0/  0 // black
```

Next, we insert a fixed node at 200 m. Now, there are two interpolation ranges, each of them from 0 % to 100 % and the fixed node at 200 m in between:

```
AT 100% R/G/B   0/  0/  0 // black
AT   0% R/G/B   0/  0/  0
FIXED NODE AT 200
AT 100% R/G/B   0/  0/  0
AT   0% R/G/B   0/  0/  0 // black
```

All colors are black. Now the color of the higher intermediate node belonging to the fixed node will be changed to red. We select the corresponding line in the text box, select a red color in the window *Color of node* and accept it with the button *Apply*. Furthermore, we change the comment of the node to "red" Now, the higher color interpolation range in the resulting ICP yields colors from red to black:

```
AT 100% R/G/B   0/  0/  0 // black
AT   0% R/G/B 255/  0/  0 // red
FIXED NODE AT 200
AT 100% R/G/B   0/  0/  0
AT   0% R/G/B   0/  0/  0 // black
```

Similarly, we insert further fixed nodes at 240 m, 260 m and 300 m resulting in five color interpolating ranges. We select the corresponding colors. The final ICP is:

```
AT 100% R/G/B   0/  0/  0 // black
AT   0% R/G/B   0/  0/  0 // black
FIXED NODE AT 300
AT 100% R/G/B 255/  0/  0 // red
AT   0% R/G/B 255/255/  0 // yellow
FIXED NODE AT 260
AT 100% I/H/S  25/120/100 // dark green
AT   0% I/H/S  75/120/100 // light green
FIXED NODE AT 240
AT 100% R/G/B 255/255/  0 // yellow
AT   0% R/G/B 255/  0/  0 // red
FIXED NODE AT 200
AT 100% R/G/B   0/  0/  0 // black
AT   0% R/G/B   0/  0/  0 // black
```

12.9. Structure

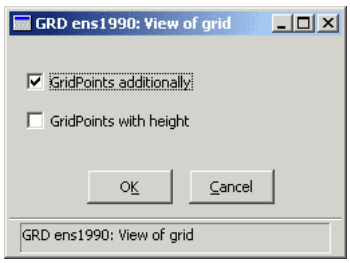


Figure 12.60.: The window *Structure*

The window *Structure* (see figure 12.60) serves for displaying the gridlines and the gridpoints in the main graphics panel. The gridlines mark the limits of the computing units. The gridpoints mark the computed raster points (see figure 5.1). You can toggle the display of the gridpoints via the checkbox *GridPoints additionally*. Mind, that the computation of the gridpoints can take a long time, especially for a large overlay. If the checkbox *GridPoints additionally* is checked the checkbox *GridPoints with height* will be accessible. Checking/unchecking this displays/removes the height of the gridpoints.

A commandline / commandfile example for this window is:

cmL example

```
ens2000/,
  Structure/,
  GridPointsAdd,
  OK;
ens2000/,
  Structure = (d);
```

12.10. Profile

This chapter describes the definition and manipulation of vertical sections of the terrain model along polygons. This is a primarily interactive module so we skip the command-line / commandfile examples.

12.10.1. Terms

Alignment: Ground plan of the profile (see below).

Profile: Alignment parametrized after the arc length, displayed in an arc length-height coordinate system.

Cross section: Ground plan of section orthogonal to the alignment. In order to define cross sections an alignment has to be defined.

Sections: Ground plans of profiles in an arbitrary number without a reference to an alignment.

This module is separated in two parts which are even on two different places in the user interface:

- The profile tool
- The extended profile facilities

The profile tool provides basic profiling ie digitization and displayal of profiles for different overlay and is situated in the menu bar of the main window in the bar item *Tool* below the entry *Profile...*. The extended profile facilities provide a large set of alignment and cross section manipulation possibilities and is only available in the module "SCOP++ Analyzer"(see section refpackages).

12.10.2. Profile Tool

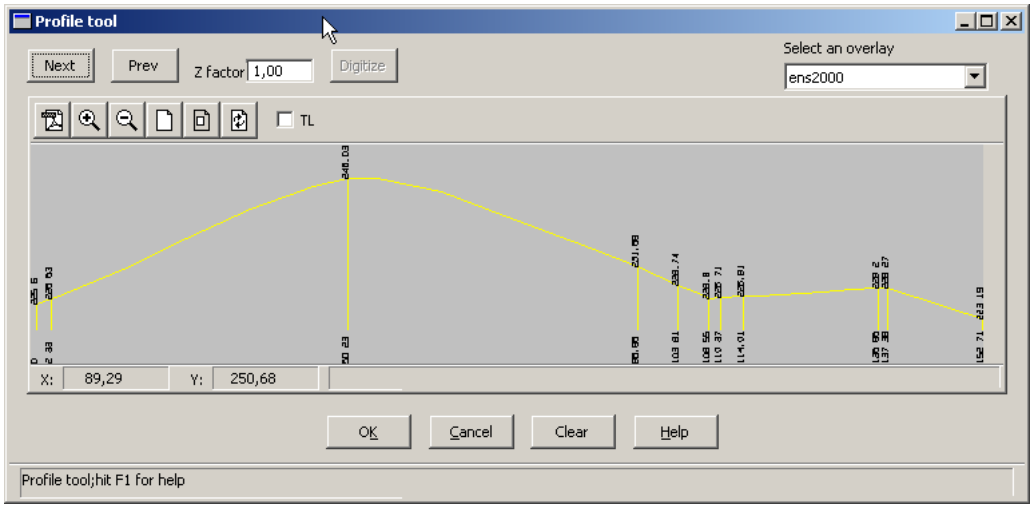


Figure 12.61.: Profile tool

This window is intended for a simple exploration of the terrain model by sections via digitizing an arbitrary number of polygons in the main graphics panel and the visualization of them. It will be locked as long as there isn't a terrain model available for at least one overlay of the project. The main graphics panel is prepared for digitization when selecting a specific overlay in the overlay selection or on opening the window *Quickview* with an overlay already selected and there is an terrain model available for it. After the termination of one polygon its profile is displayed in the graphics window of this window. The profiles of recently digitized polygons can be redisplayed via the button *Prev* and the button *Next*. The section whose profile is currently depicted in the graphics area of this window is highlighted (red) in the main graphics panel. Closing the window *Quickview* either on pressing the cross in the upper right corner or selecting the Off state of the state-switch *Profile ...* will discard all digitized sections. They will be redisplayed when opened again. If there are already profiles digitized and another overlay is selected it will be tried to recalculate the profiles in the selected overlay. Doing this the profiles may be clipped resp. skipped if they aren't contained within the extents of the selected overlay.

The window will be closed if any other process tries to access the DTM file (eg. recalculation of the DTM or calculation of contourlines), because the DTM file can only be accessed once.

If another module sets the main graphics panel into digitizing or editing state the window *Profile tool* will be closed and has to be reopened again if needed.

12.10.3. The Profile Window

This window (see figure 12.62) is the main window of the profile view. With the input fields of this window the alignment and the cross sections (optionally) are defined and manipulated.

Alignment Definition There are a few caveats concerning alignments:

- Be aware that the alignment will always be treated as straight line (in contrast to the `SCOP V3.5` module PRO).
- The alignment will be clipped if parts of it aren't defined in respect to the DTM (ie. if the alignment crosses areas where the DTM isn't defined).

The selection which way of definition will be chosen must be made in the group *Input device*. Via checking one of the radio buttons *Cursor, Keyboard, From file* this selection is made. Checking the radio button *From File* unlocks the button *File...* with which a file dialog can be opened. The alignment can be defined in three different ways:

1. Digitizing the alignment in the main graphics panel. Checking the radio button *Cursor* prepares the main graphics panel for digitizing the alignment. You can then bring the window *Scop++ main* to the foreground and digitize an alignment. Having completed the alignment by pressing the right mouse button, the profile will be calculated and displayed in the window *Profile* (see figure 12.62).

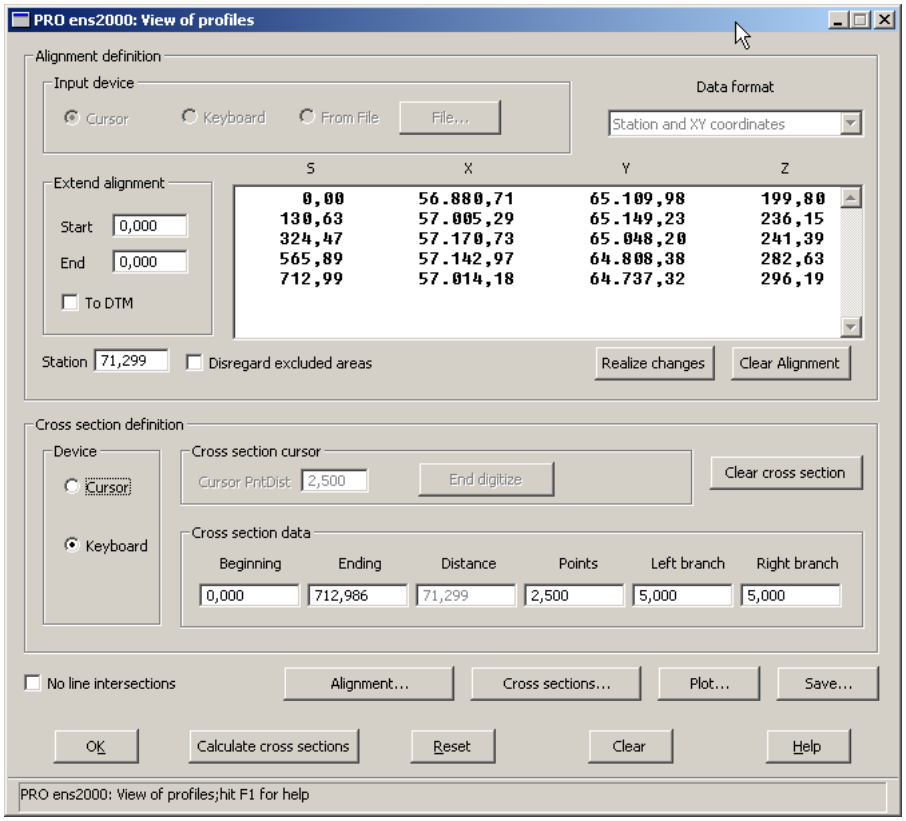


Figure 12.62.: Profile

2. Entering in the X- and Y-coordinates of the alignment manually. The selection of the radio button *Keyboard* in the group *Input device* makes the text window *Alignment data* accessible. Besides the selection *Format* will be set to the item Only XY-coordinates. If you want to enter the station of the alignment points also, please select the item Station and XY-coordinates in the selection *Data Format*. You can enter the coordinate directly into the the text window *Alignment data*. After pressing the button *Realize changes* the alignment will be displayed in the main graphics panel, the profile will be calculated and the coordinates of the alignment points (station x y z) will be displayed in the text window *Alignment data*.
3. Reading in the X- and Y-coordinates of points defining the alignment. After you have selected the radio button *From File* in the group *Input device* the button *File...* is unlock. On an open a file input dialog is displayed. In the text field *Name* you can enter the path and the name of the file with the coordinates manually or you can browse for them with a file-browser which can be open by pressing the button *Browse...*. Be sure you have made the right specification for the data format in the group *Data Format* (see figure 12.62). It is supposed the contents of the file matches the format used when saving an alignment (see section 12.10.6), but the keyword `Alignment` followed by the number of alignment points also can be skipped. In this case it is assumed that all the points on the file belong to the alignment.

Alignment Manipulation The alignment can be manipulated in two different ways:

1. Altering the X- and Y- coordinates of the points of the alignment in the text window *Alignment data*.
2. Interactively alter the alignment in the main graphics panel with the standard means (see section 8).

Furthermore the depiction of the alignment can be manipulated:

1. State the distance of points calculated along the alignment. This can be done via the numeric field *Station*. Beware that this parameter is used for the plot of the alignment and the calculation of the cross sections (see section 12.10.3) only.
2. Disregard excluded areas within the DTM via checking the checkbox *Disregard excluded areas*.

Extend Alignment For some application areas it might be worthwhile to have the possibility to extend the alignment either up to a specified amount or to the border of the DTM. This can be accomplished by the means of the group *Extend*. Checking the checkbox *To DTM* will extend the alignment to the DTM, whereas entering values into the numeric field *Start* or the numeric field *End* will extend the alignment by the specified value.

The ground plan of the alignment is displayed in the main graphics panel and its profile will be displayed in the window *View alignment* (see section 12.10.4).

Cross Section Definition The Cross sections can either be calculated automatically or be digitized with the cursor in the main graphics panel.

1. Digitizing the cross sections

   If they should be digitized the radio button *Cursor* in the group *Input device* (see figure 12.62) must be selected. Following that, points can be digitized in the main graphics panel, from which the cross section will be calculated and displayed perpendicular to the alignment. The cross section will be extended to the same distance as the digitized point on the other side of the alignment. Along the cross sections the height is determined in a distance which can be defined in the numeric field *Cursor PntDist* in the group *Cross section cursor*. In the same group the button *End digitize* is situated, with which the digitizing state can be quitted.

2. Providing the cross sections automatically

   The cross sections (their profile and position) can be triggered by selection of the radio button *Keyboard*. In the numeric field situated in the group *Cross section data* right from it, the parameters for the cross sections can be manipulated. The following numeric fields will be available:

   Beginning: Station of the alignment where the automatical computation should start. Is zero per default (the computation starts at the beginning).

   Ending: Station of the alignment where the automatical computation should stop. Equals the length of the alignment per default (the cross section will be calculated for the whole alignment).

Distance: Distance of the cross sections in direction of the alignment. This value will be set to one tenth of the length of the alignment per default. Furthermore it isn't accessible for the user at all, because it corresponds the 'Station' value of the alignment (see above).

Points: Distance of the points on the cross sections where a height should be calculated. Is 2.5 meter per default.

Left branch: Length of the left branch of the cross section (the direction of the alignment is defined by the order of the points which define it).

Right branch: Length of the right branch of the cross section.

The automatic calculation of the cross sections is triggered by pressing the button *Calculate cross sections*. The ground plan of the cross sections is displayed in the main graphics panel and their profiles will be displayed in the window *Cross sections* (see section 12.64). The cross sections will be calculated on pressing the button *OK* also - if they were defined previously - but this window will be closed immediately afterwards. The cross sections can be deleted by pressing the button *Clear cross section*. If the button *Clear* in the last row of this window is pressed all specifications will be discarded.

The button *Alignment...* Behind this button there is the window for viewing the alignment(see section 12.10.4).

The button *Cross sections...* Behind this button there is the window for viewing the cross sections (see section 12.10.5).

The button *Plot...* Behind this button there is the window for generation of plotfiles according to the `SCOP V3.5` directives LONGPLT and CROSSPLT (see section 12.10.7).

The button *Save...* Behind this button there is the window for saving the alignment and cross section definitions to file (see section 12.10.6).

Data Formats of the `SCOP V3.5` Profile Module In the `SCOP V3.5` Profile Module different data formats for the alignment and the cross sections are used:

For the alignment data the data format DA040 was used.
The alignment points are read in using the format

```
(I3,I2,F10.3,F8.3,F12.4,F9.3,F12.7,2F12.3)
```

Column	Format	Contents
1-3	I3	`040`
4-5	I2	Number of alignment, must be the same for one alignment set
6-15	F10.3	Station of alignment point
16-23	F8.3	Difference to station of previous alignment point, zero for the first point
24-35	F12.4	Radius of curvature in alignment point (+ = right hand bend, − = left hand bend, zero = no curvature (straight line or klothoide beginning with infinite radius of curvature))
36-44	F9.3	Parameter of klothoide; (+ = increasing curvature, − = decreasing curvature, zero = constant curvature (circle or straight line))
45-56	F12.7	Angle [gon] of the tangent in the alignment point to the north axes, clockwise.
57-68	F12.3	East coordinate of alignment point
69-80	F12.3	North coordinate of alignment point

A left hand bended alignment element with increasing curvature, beginning with radius zero, needs a radius of −0.0001.

For the station data the data format DA055 was used.
The format of station data records is:

```
(I3,I2,F10.3,5(F6.2,F7.2))
```

Column	Format	Contents
1-3	I3	`055`, `056`, `057` or `058` any of these values may be used, however it should be the same for one computation and it should correspond to the last value given in the range data.
4-5	I2	Station card number; counting from -1 to -7 for points left of the alignment, and from 1 to 7 for points right of the alignment. One cross section may not contain more than 35 points on either side of the alignment.
6-15	F10.3	Station of the cross section.
16-21	F6.2	Distance from the alignment for point 1, to the left, if station card number negative, to the right, if positive.
22-28	F7.2	Height for point 1.
29-34	F6.2	Distance for point 2.
35-41	F7.2	Height for point 2.
42-47	F6.2	Distance for point 3.
48-54	F7.2	Height for point 3.
55-60	F6.2	Distance for point 4.
61-67	F7.2	Height for point 4.
68-73	F6.2	Distance for point 5.
74-80	F7.2	Height for point 5.

12.10.4. The Alignment View Window

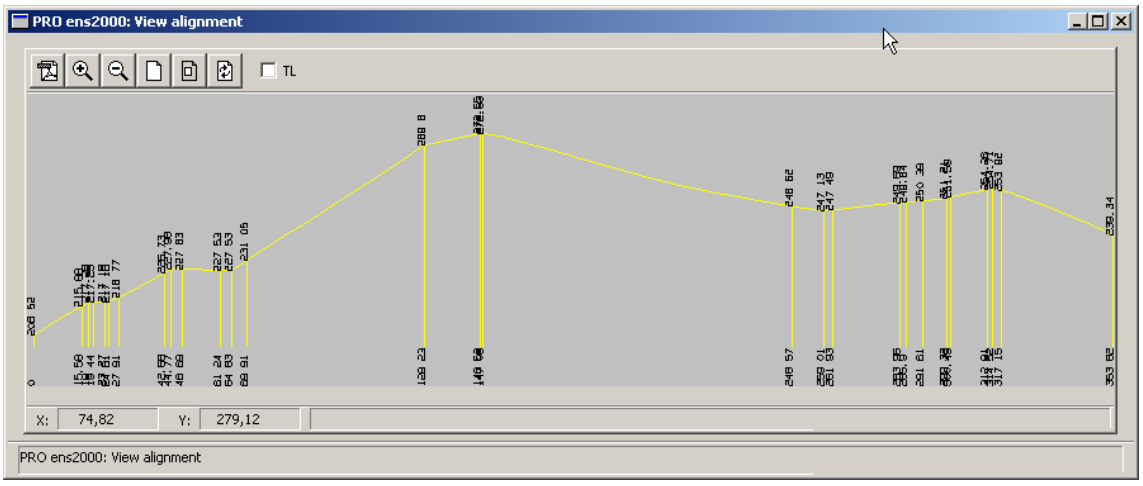


Figure 12.63.: Alignment view window

In its graphic area the profile will be visualized, if an alignment is defined. The displayed profile contains the following information:

- The profile
- Arc lengths: For the digitized points, for crossings of the alignment with breaklines, and for crossings with triangulation edges within a raster cell . For all these points a vertical line is displayed. The arc lengths are written on the lower end of the vertical lines.
- Heights: For all points, for which an arc length is displayed, the height is also displayed (on the upper end of the vertical line).

12.10.5. The Cross Section Window

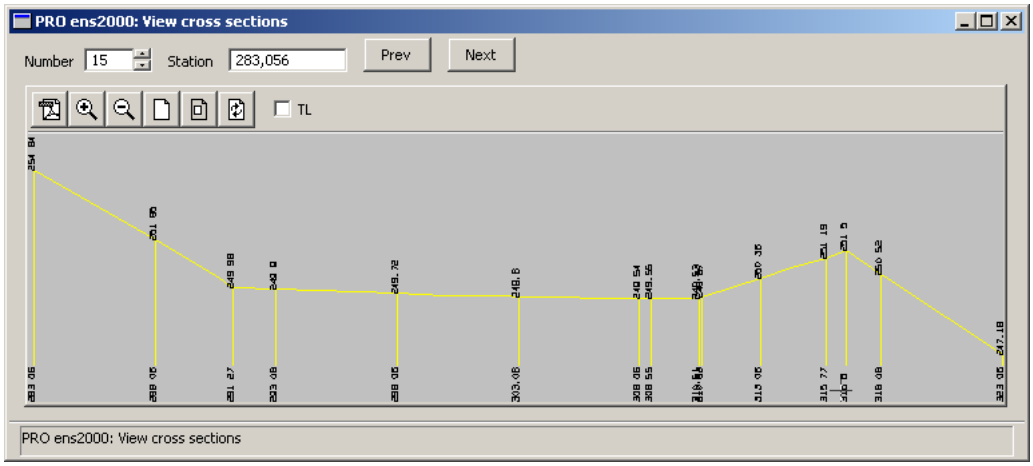


Figure 12.64.: The window *View cross sections*

In its graphics window the cross sections, if defined previously are displayed. From the set of defined cross sections a special one can either be chosen via its number (enter it in the numeric field *Number* in the top left of the window) or with the button *Prev* and the button *Next*. Latter show the previous resp. the next cross section in the graphics window. To ease the identification which cross section is currently displayed, the corresponding ground plan is highlighted (red) in the main graphics panel. In the numeric field *Station* on the top of the window the station of the displayed cross section can be examined. Similar to the representation of the profile on the upper end of the vertical lines the height of the point is drawn and at the lower end of the vertical line the arc length of the point in the cross section. It starts at zero at the left end of the cross section which is determinated via the direction of the alignment (see above).

12.10.6. Save Profiles

Via this window the planimetric data of the alignment and the cross sections can be saved to a file in various formats. The available formats are:

AutoCAD DXF: Drawing Interchange File Format of Autodesk, ASCII

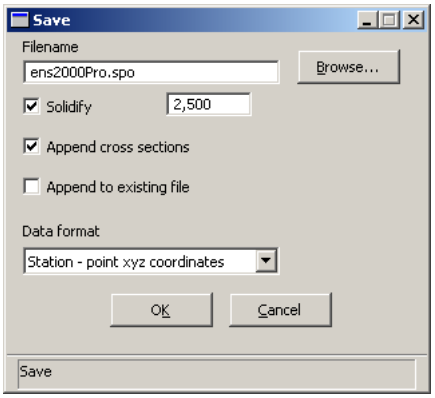


Figure 12.65.: The window *Save*

SCOP WINPUT: Proprietary intermediate file format of SCOP, ASCII. The profiles and cross sections - if available - are stored as form lines.

XYZ: XYZ coordinates of points

Station-point xyz coordinates: Station and XYZ coordinates of points. An alignment will be started with the keyword alignment which is followed by the number of alignment points. After the alignment the cross sections will be listed. Each cross section will be started with a line containing the station of the cross section and the number of points of the cross section. For example:

```
Alignment 226
       0.000    57279.720    65037.609      222.289
       2.500    57277.224    65037.747      223.282
       5.000    57274.728    65037.886      224.276
       ....

Cross sections
Enter 0.000 353.824 20.000 2.500 20.000 20.000
12.092 22
       0.000    57268.756    65058.248      226.346
       2.500    57268.618    65055.752      226.374
       5.000    57268.479    65053.256      226.436
       ....
```

12.10.7. Plot Profiles

In this window (see figure 12.66) the parameters of the `SCOP V3.5` LONGPLT directive can be accessed. This serves for generating an plotfile as some users have get used to.

Profile Output In this group specifications concerning the resulting file can be made. In the text field *FileName* the name of the plotfile can be determined. If the checkbox *Append* is selected the new plot instructions will be appended to an file if it yet exists. Otherwise

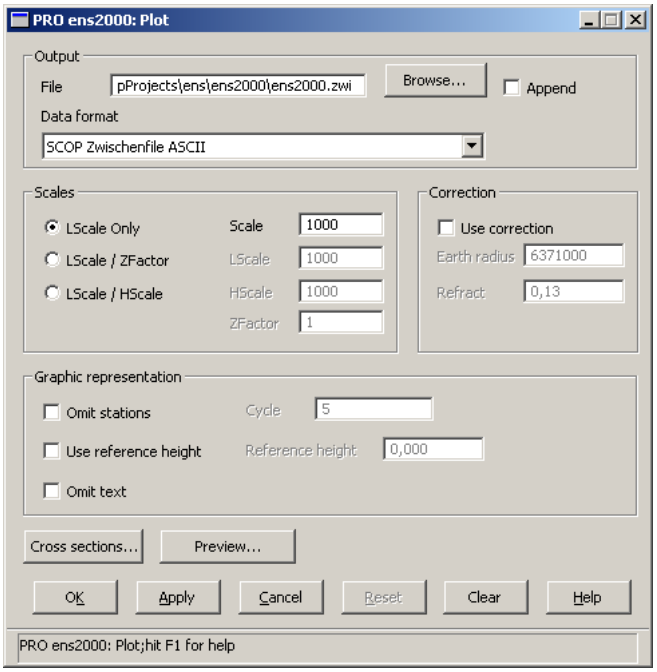


Figure 12.66.: The window *Plot*

an existing file will be overwritten. The selection *Data format* serves for choosing the output format. The supported formats are:

Supported output formats
HPGL Hewlett Packard Graphics Language
SCOP Zwischenfile ASCII
AUTOCAD DXF

Scales In this group the scale of the plotfile can be accessed. On its left margin there are three exclusive radio buttons :

LScale Only states that only one scale factor can be defined. This factor will be applied to both axes of the plot.

LScale/ZFactor states that there can be defined a scale for the arc length axis and a multiplication factor for the z axis.

LScale/HScale states that there can be defined a scale for the arc length axis and a scale for the z axis.

In the numeric field right to the radio buttons these values can be manipulated. These numeric fields are locked resp. unlocked according to the chosen radio button .

Corrections These parameters may be used to correct refraction and earthcurvature for very long profiles that are straight lines (only a beginning and an endpoint). The checkbox *Use correction* decides whether or not to apply the correction. According to its state the other input fields in this group are locked resp. unlocked. In the numeric field

Earth radius a value for the radius of the earth and in the numeric field *Refract* a value for the refraction can be entered.

Graphic representation Via the input fields situated in this group you can define the appearance of the plot. The state of checkbox *Omit text* decides whether the profile should be plotted with a legend and labelling of stations, scale and height or not. The checkbox *Omit stations* decides if only every i-th station defined in the numeric field *Cycle* should be plotted and labelled with a height. The numeric field *Cycle* is locked resp. unlocked according to the status of the checkbox *Omit stations*. The numeric field *RefH* defines a reference height for the plotted heights.

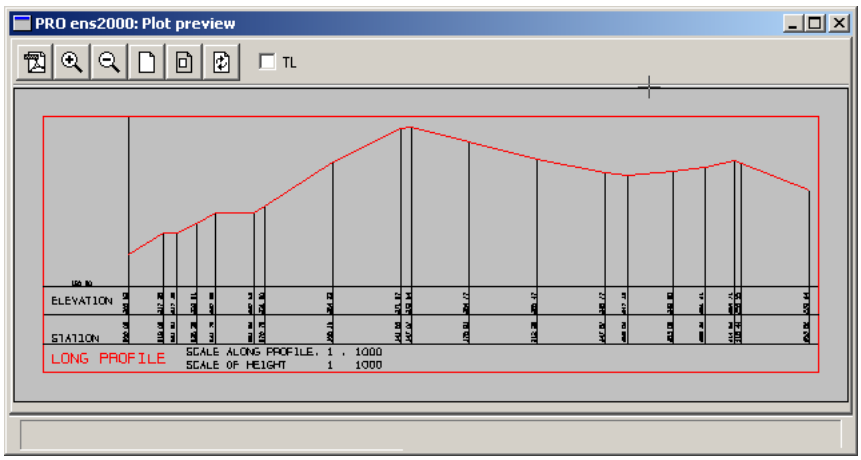


Figure 12.67.: The window *Plot preview*

The state-switch *Preview* The state-switch *Preview* opens the window *Plot Preview* (see figure 12.67) consisting only of one graphics area, which serves for a plot preview, if the state-switch is in Display state.

The button *Cross section...* The button *Cross section...* opens the window *Plot cross sections* (see figure 12.68). This button will be locked if the cross sections aren't determined automatically (see section 12.10.3).

12.10.8. Plot Cross Sections

In this window (see figure 12.68) the parameters of the `SCOP V3.5` CROSSPLT directive can be entered.

Output This group corresponds in its function with the one of the window *Plot*(see section 12.10.7).

Scales The input fields correspond to the ones of the group *Scale* of the window *Plot*(see figure 12.10.7).

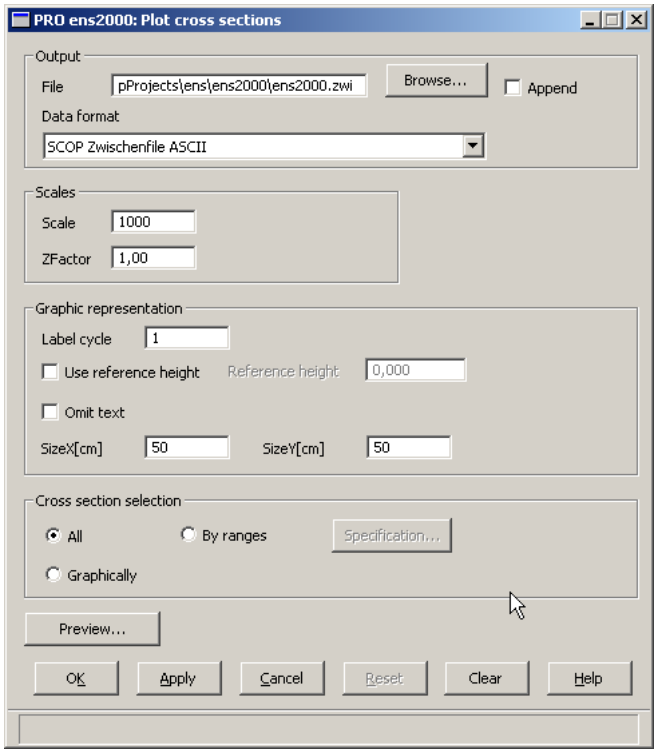


Figure 12.68.: The window *Plot cross section*

Graphic Representation In the input fields of this group the appearance of the plot of the cross sections can be determinated. The numeric field *Label Cycle* determines the cycle of full labeling of the cross section in the plot. The state of the checkbox *Omit text* decides if any cross section at all should be provided with a text (caption of the arc length axis and height). The numeric field *SizeX[cm]* and the numeric field *SizeY[cm]* states the extensions of the plot. Cross sections which come to lie outside of this plot extensions are skipped. The checkbox *Use reference height* together with the numeric field *Reference height* allows to specify a reference value for the plotted heights.

Cross Section Selection Via the input fields of this group a selection which cross sections should be plotted can be made. There are three possibilities:

All All cross sections will be plotted (Mind the restrictions due to the plot extensions).

Graphically If this item is selected the cross sections, which should be plotted can be selected in the main graphics panel. The selected cross sections will be highlighted (red).

By ranges Selecting this item, unlocks the button *Specification...*. Via the window behind this button ranges, referring to the arc length of the alignment, can be specified within those the cross sections will be plotted (see section 12.10.9).

The state-switch *Preview* The window opening behind this button corresponds in its function with the one of the window *Plot* (see section 12.10.7).

12.10.9. Range Specification

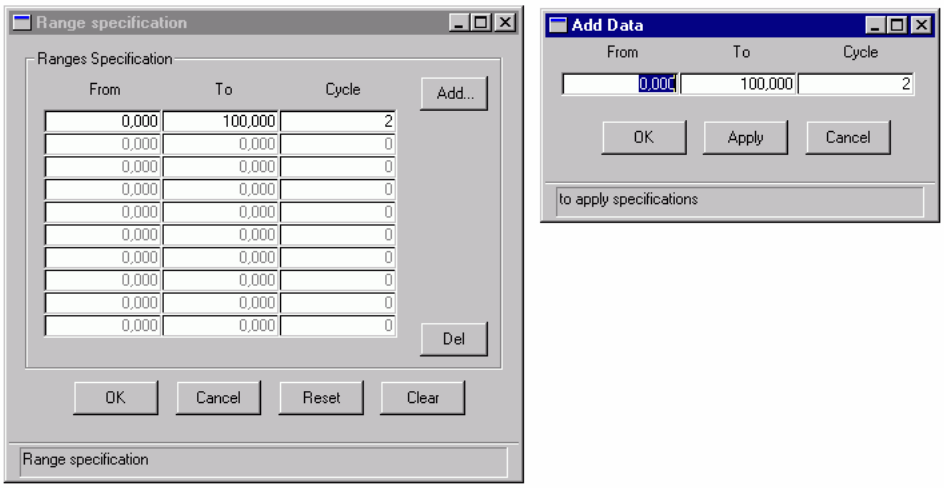


Figure 12.69.: Range specification

In this window (see figure 12.69) ten triples of values concerning the constitution of the set of cross sections which will be plotted. The first column of numeric fields *From* determines the arc length on the alignment, from where on the cross section will be plotted. The second column of numeric fields *To* determines the arc length on the alignment, up to which the cross sections will be plotted. With the third row of numeric fields *Cycle* you can determine a subset of the cross sections which will be plotted. For example if one row comprises the values 100.000, 200.000 and 2 will state that starting from station 100.000 up to the station 200.000 of the alignment each second cross section will be plotted. You can easily check your specifications in the window *Preview* (see section 12.10.8). You can add value triples by pressing the state-switch *Add* and entering the desired values in the numeric fields in the appearing window. If you press the button *Apply* the values will be added; if you press the button *Ok* the window will be closed additionally. Pressing the button *Del* will delete the very triple which has the focus.

12.11. Slope

This chapter describes the calculation and simple visualization of slope models. As all other views dialogs this dialog isn't capable of creating a product (slope model) for external use. For this task please refer to the Import/Export section (see section 12.2.8). The visualization capabilities are restricted to a simple Z-coding. In order to exhaust the whole extent of the visualization parameters export the slope model and import it as a model-only overlay.

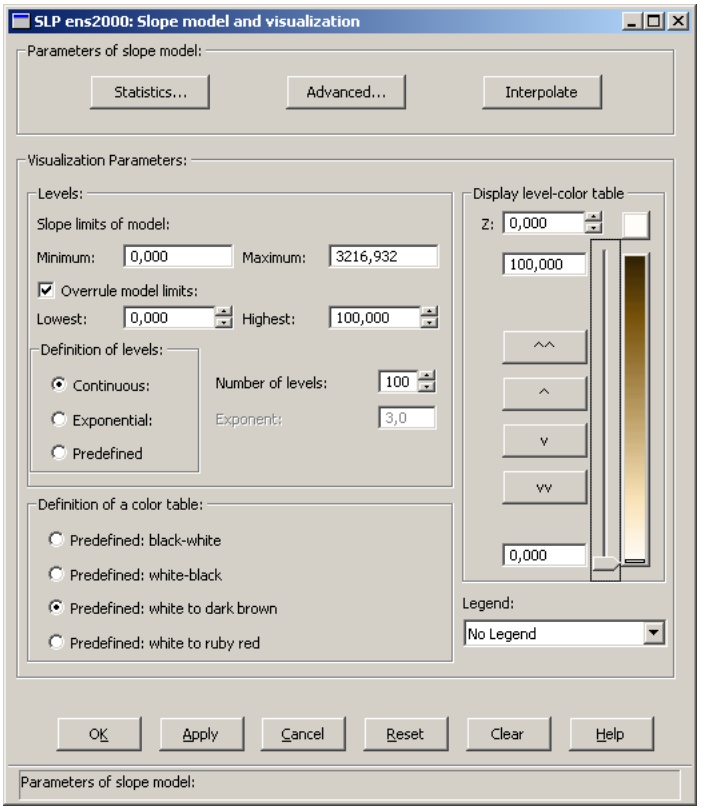


Figure 12.70.: Slope window

This is a special kind of view because it's necessary to calculate a secondary model (the slope model) of the primary model (the DTM of the overlay) first. The structure (e. g., the grid size etc.) of the slope model is the same as the structure of the primary model, but the computed values are steepness values (slopes in [%]). For the computation of the slope in a grid point, a moving tilted plane defined by the surrounding grid and line points is used. The slope in the grid point is then defined as the maximum slope of this plane.

At break-line points the slope is computed on each side of the line. To store the two slope values it is necessary to double the break-line. Thus after the computation, each break-line occurs twice. The two lines have identical eastings and northings but different slope values.

This window offers not only the possibility to calculate a slope model, but also to visualize it by means of a Z-coding (i. e. generating a slope-coded image). On the GUI, the

combination of secondary model and simple visualization is presented as state-switch with two lights: a round light (indicating the state of the secondary model) and a rectangle light (indicating the visualization state) (cf. section 5.1.2). Both parameter sets, the model and the visualization parameters, can be selected on one common parameter window.

12.11.1. Parameters for the Secondary Model

The group *Parameters of slope model* comprises:

- Parameters for output of statistical information. (see section 12.11.2).
- Advanced parameters for the secondary model (see section 12.11.3).
- An interpolate button .

A new derivation of the secondary model will only occur, when it is immediately needed for some other process – e.g. if the state-switch *Slope* is set to *display* or the primary model has changed and the state-switch *Slope* is in *display* state. However a new derivation can be forced by pressing the button *Interpolate*.

After the slope model has bee derived, the model part of the state-switch *Slope* (see section 5.1.2) will have a yellow light on it.

12.11.2. Statistics about the Slope Model

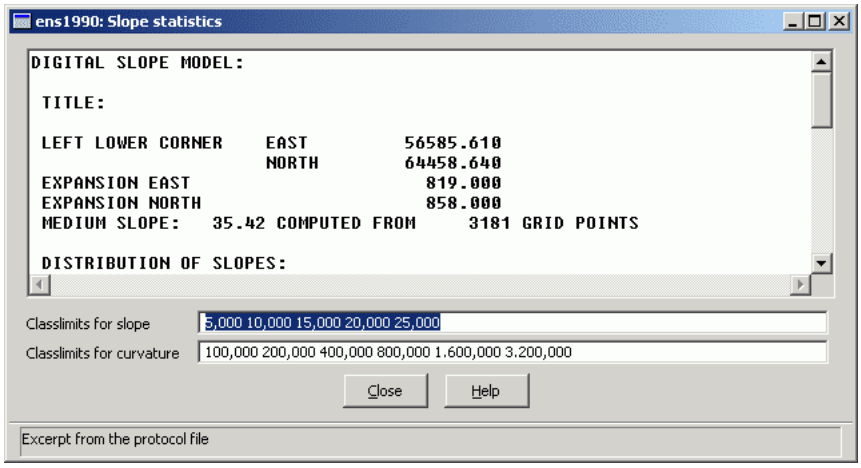


Figure 12.71.: Slope statistics

After the computation of the Digital Slope Model, the average slope and curvature as well as some statistics about slopes and curvatures are listed. These statistics are visualized in the text window of dialog window *Slope Statistics*.

The classlimits for slope and curvature can be entered in the corresponding text fields . These values must be given in ascending order and the lower limit is always zero and will not be listed. Be aware that there is an upper limit of 20 classes for both of them.

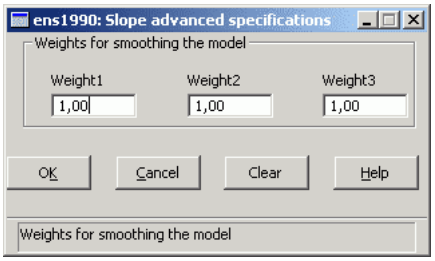


Figure 12.72.: Slope statistics

12.11.3. Advanced Parameters for the Secondary Model

These are three weight parameters intended for smoothing the resulting model. The slope model is smoothed by averaging the slopes of the surrounding grid points and grid intersections. The grid points and grid intersections can be weighted depending on their distance to the computed point. In addition, a weight for the computed point can be given. That way, different smoothing effects can be reached. The first value is the weight of the point to be computed. The second value is the weight of a surrounding point in minimum distance. The minimum distance is assumed to be zero. The third value is the weight of a surrounding point in maximum distance. The maximum distance is the diagonal of the grid. The weight of a point is then computed as a linear function of the second and third value.

12.11.4. Parameters for Visualization

For a simple visualization of the slope model, a Z-coding is used. The parameters are a subset of the parameters of the "main" Z-coding parameter window (cf. section 12.8). The predefined palettes are specific ones. Additionally there is a special entry *Predefined* within the group *Definition of levels*. When selecting *Predefined* the class limits from window *Statistics...* will be used as level definitions to get a corresponding visualization.

To benefit of the full potential of the Z-coding abilities under SCOP++ you have to export the slope model and insert it as a model-only overlay (see section 12.2.8).

12.12. Quality

DTMs are used for various purposes. If a decision is reached on the basis of a DTM, the quality of the decision depends on the quality of the DTM — or the fitness of the DTM for this purpose. Therefore it is inevitable to provide DTM quality measures within SCOP++.

Quality models in SCOP++ are regarded as secondary models in contrast to the primary model (i.e. the DTM). They can be made persistent either as own models (Z-value of the grid-point represents the quality measure), or by importing the quality measures into an existing DTM (see section 12.2.4).

The quality dialogs are accessible through the window *Model Overlay* (see section 12.1). In this window the quality-buttons will unfold resp. fold on pressing the button *Quality* (see figure 12.73).

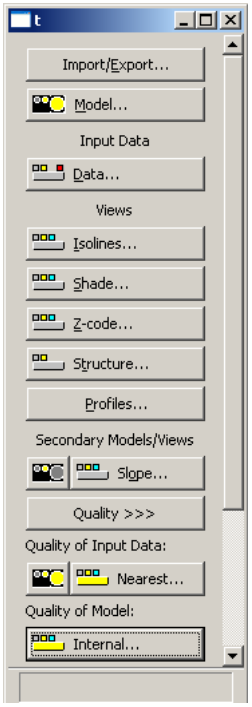


Figure 12.73.: Model Overlay window with unfolded Quality

12.12.1. Best Neighboring Data Class

A data class map shows for each pixel centre the class of the most accurate data inside the area covered by the pixel representing an analysis unit. This is based an the assumption that the most accurate data point near an interpolated grid point will have the highest influence on the accuracy of this grid point. Therefore in this method, a regular grid of analysis units is laid over the whole *Model* area. Subsequently, each analysis unit is classified according to the best quality class of the data in the unit. The dialog window *Nearest quality window* (see figure 12.74) provides the manipulation of the parameters for the secondary model as well as the parameters for a simple visualization.

Parameters for the secondary model The group *Parameters of nearest reference point model* comprises the parameters for the secondary model: Parameters determining

1. the structure and
2. the values

of the resulting model.

The structure of the resulting model is defined in the group *Definition of analysis unit AU*. The size of the analysis unit is the grid size of the resulting model. It can be defined in the numeric field *AU size*. The number of analysis units may be set in the numeric fields *Number of AUs*, both in East and North direction. These values can be chosen arbitrarily. However, in order to import the resulting model into the existing primary model, these values must not be altered, at best by leaving the checkbox *Overrule unit size* unchecked. Additionally, it is then guaranteed that changes of the structure of the primary model are immediately reflected in changes of the structure of the secondary model.

The mapping of the available *featurecodes* (see 5.3.10) to quality classes is defined in the lower part of the group *Parameters of nearest reference point model*. By default, all available

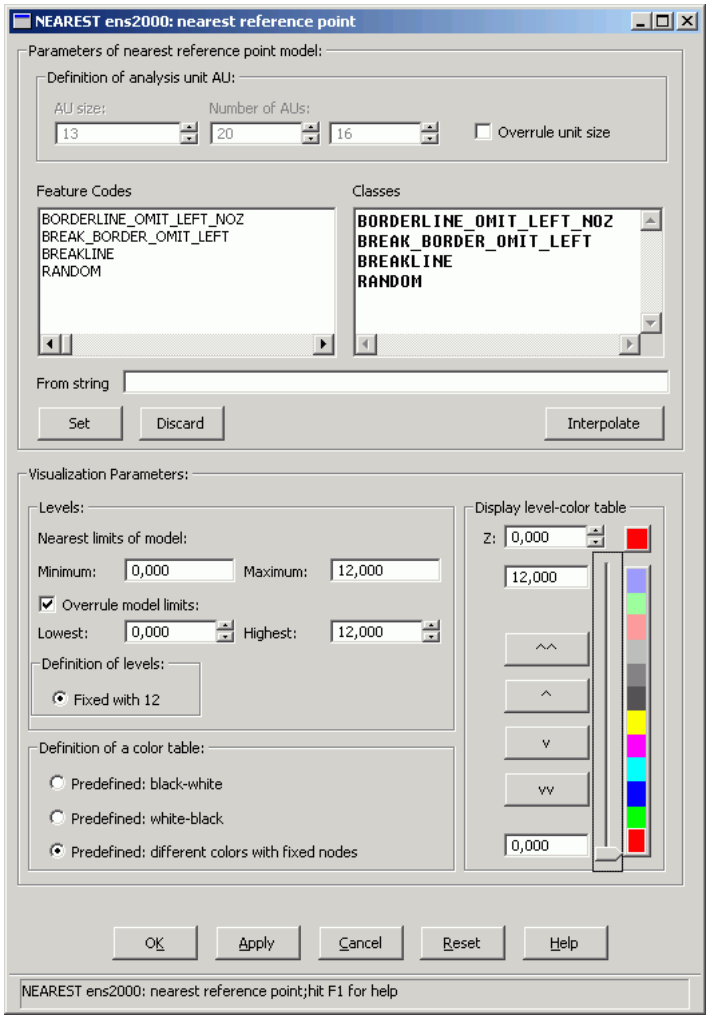


Figure 12.74.: Nearest quality window

featurecodes are associated with a different quality class, in alphabetical order. However, as the order is essential, this setting should be changed in most cases. In order to do so, the defaults must be deleted first by pressing the button *Discard*. Subsequently, new quality classes must be defined, this time in a meaningful order and eventually with several *featurecodes* of same quality being merged into a single quality class. Quality class definition is done either by

- selecting one or more *featurecodes* in the selection *Feature Codes*, or by
- typing in one or more *featurecodes* in the text field *From string* (separated by blanks, commas, or semicolons)

and then pressing the button *Set*. Mind that each *featurecode* can only be part of one class.

Each quality class is represented by one line of *featurecodes* in the text window *Classes*, separated by blanks. The higher a quality class is situated in the list of window *Classes*, the better the quality it represents. The order of the quality classes is essential, because each analysis unit is assigned the best quality occurring inside the analysis unit.

Although the (re)calculation of the nearest model will be performed according to the principle of *lazy processing* (see section 5.3.1) a calculation can be forced by hitting the button *Interpolate*.

Parameters for visualization For a simple visualization of the Nearest model, a Z-coding is used. The parameters are a subset of the parameters of the "main" Z-coding parameter window (see section 12.8). The predefined palettes cannot be altered.

The color table is restricted to 12 levels because this is the maximum number of classes which can be imported into a DTM. The three predefined color tables provide adapting, greyish (white-black, black-white) as well as coloured visualization.

In order to benefit from extended visualization possibilities, the nearest model must first be exported and then imported into a (newly created) model-only overlay (see section 12.2.8 and section 12.2.4).

A commandline / commandfile example for this window is:

cmL example

```
@Nearest;
  ens2000/
  "Quality>>>",
  Nearest,
    Discard,
    BORDERLINE_OMIT_LEFT_NOZ, BREAK_BORDER_OMIT_LEFT,
    Set,
    BORDERLINE_OMIT_LEFT_NOZ, BREAK_BORDER_OMIT_LEFT,
    // deselect the selected, because the feature-codes must
    // be of unique occurrence.
    BREAKLINE, RANDOM,
    Set,

    OverruleUnitSize=(n),
    // Use the same value for unit size as used for
    // DTM computation, in order to enable the import of
    // the quality into the primary model

    Predefined:differentColors,
    ok;
  Nearest=(d);
@endProc;
```

12.12.2. Point Distance

The information about the 2D-distance of a grid point to the next available data point is a valuable parameter for estimating the input data quality. Large distances often indicate occlusion areas (e.g. scan shadows) or impenetrable areas (e.g. ground point gaps after filtering of laser scanning data) and can lead to artifacts in the derived DTM surface. Within SCOP++, distance models are calculated based on the Chamfer Distance Transform (CDT) algorithm known from image processing. The CDT offers a good approximation to the real (Euclidean) distances at a lower computational cost. Figure 12.75 shows a schematic diagram illustrating the assignment of distances to the single grid point.

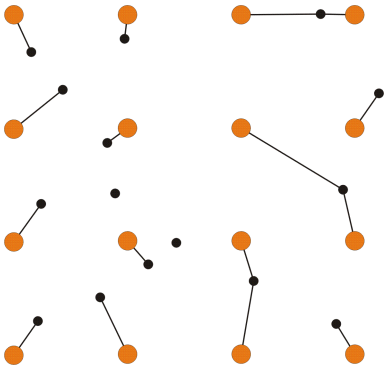


Figure 12.75.: Assignment of chamfer distance to grid point

For each grid point, the nearest data point is retrieved and the (approximative) distance, as calculated by the CDT algorithm, is assigned to the grid point. As can be seen, the very same data point can be linked to more than one grid point in case the respective data point is located nearest to multiple grid points.

The dialog window *Distance to nearest data point* (see figure 12.76) provides the manipulation of the parameters for the distance model as well as the parameters for a simple visualization (distance map).

Calculating the distance model (as a secondary model) only requires the size of analysis units (AU). Please refer to section 12.12.1 for a detailed description concerning the manipulation of the AU size. Furthermore, the dialog provides parameters for deriving a simple visualization (distance map) as a Z-Coding with predefined grey palettes (black-white, white-black) and continuous or exponential level definitions. For exploiting the full Z-Coding visualization possibilities, the distance model can be exported as a secondary model and imported as an independent model overlay.

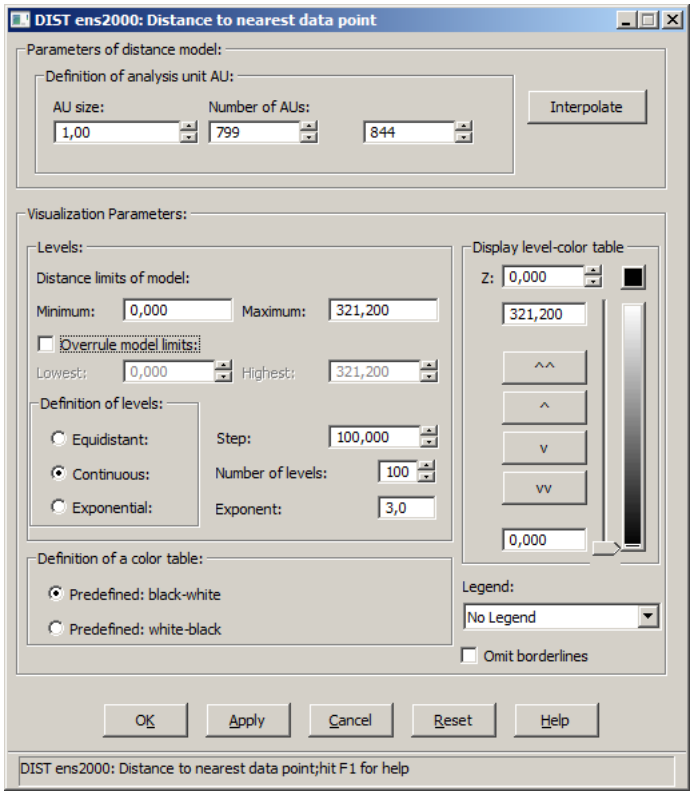


Figure 12.76.: Distance to nearest data point window

A commandline / commandfile example for this window is:

cmL example

```
@Nearest;
  ens2000/
  "Quality>>>",
  Distance,
    // Distance model parameters
    AUSize=(5.0),
    // Visualization model parameters
    Predefined:black-white,
    DefLevels=Continous,
    NrOflevels=100,
    ok;
  Distance=(d);
@endProc;
```

12.12.3. Point Density

Point density maps are widely used to check the completeness of a data capture (coverage). Such a layer is especially useful for laser scanning projects, where different parameters like flying altitude, surface roughness, soil moisture content, etc. influence the amount of recorded echoes.

As for the Nearest and Distance Model, a regular grid of analysis units is laid over the entire *Model* area. All data points of a respective cell are counted and divided by the cell area (cf. figure 12.77). This quotient is assigned to the grid cell. Thus, each cell contains a density measure in $[points/m^2]$ (in case of metric data). By applying an appropriate color table, the derivation of a density map becomes simple, allowing a quick and convenient way for inspecting the coverage of a data capturing mission.

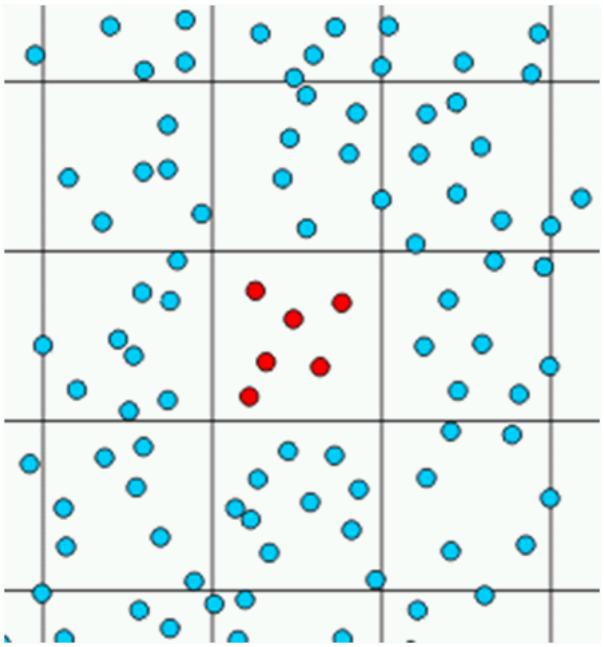


Figure 12.77.: Density model, schematic diagram

The dialog window *Point density* (cf. figure 12.78) provides the manipulation of the parameters for the density model as well as the parameters for a simple visualization (density map).

The derivation of the point density model only relies on the size of analysis units (AU). Please refer to section 12.12.1 for a detailed description concerning the manipulation of the AU size. A simple raster graphic visualization of the point density can be derived featuring predefined grey or colour palettes and continuous or exponential level definitions. Again, for exploiting the full Z-Coding visualization possibilities, the density model can be exported as a secondary model and imported as an independent model overlay.

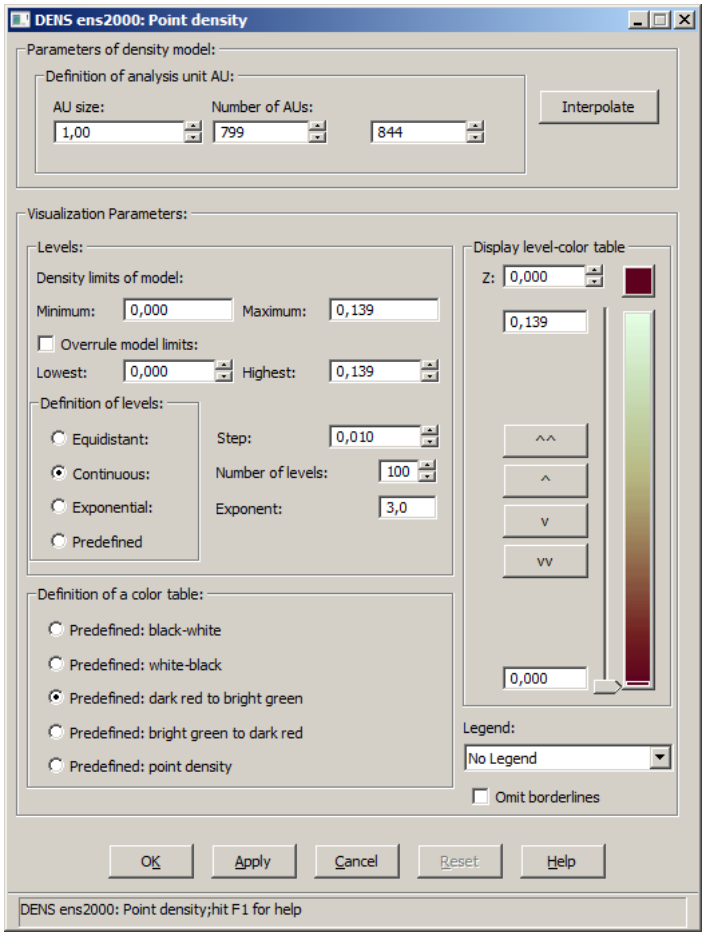


Figure 12.78.: Distance to nearest data point window

A commandline / commandfile example for this window is:

cmL example

```
@Nearest;
  ens2000/
  "Quality>>>",
  Density,
    // Distance model parameters
    AUSize=(5.0),
    // Visualization model parameters
    Predefined:brightgreentodarkred,
    DefLevels=Continous,
    NrOflevels=100,
    ok;
  Density=(d);
@endProc;
```

12.12.4. Visualization of the Internal Quality of the DTM

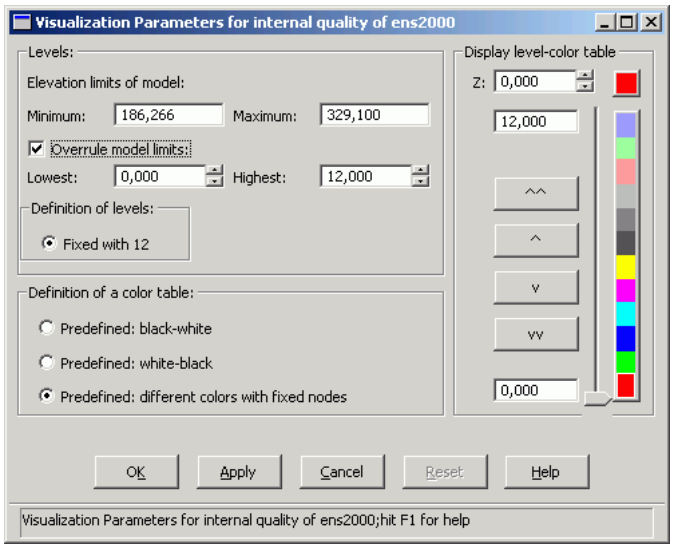


Figure 12.79.: Internal quality window

Quality measures which are stored within the primary model are called internal quality (see section 12.2.9). The window *Visualization Parameters for internal quality* (see figure 12.79) serves for the visualization of the internal quality of the DTM. It follows the functionalities of the visualization group of the window *Quality as Distance to Nearest Reference Point* (see section 12.12.1).

A commandline / commandfile example for this window is:

cmL example

```
@Internal;
  ens2000/
  // Export the nearest model.
  ImportExport,
    SecondaryModels,
    Export,
      Filename=(ensNearest),
      ModelToExport=(Nearest),
      ok;
    close;

  // Import the nearest model as internal quality.
  // Only possible, if there is a DTM and ifthe nearest model's
  // unit size and the DTM's grid size are equal!
  ImportExport,
    "Secondary Models",
    Import,
      Filename=(ensNearest),
      Type=(1:1),
      ok;
    close;

  // Now display the internal quality
  Internal=(d);

@endProc;
```

13. Image Overlays

Image overlays allow to add a digital image to the main graphics panel. Read section 5.3.8 to learn more about the benefits of adding image overlays.

13.1. Adding a New Image Overlay

In the list of images which make sense to add to a project (cf. section 5.3.8), we can see that either – in most cases – images are imported from an external image data file or – as it is the case in the last item of the list – the image is created in a model overlay of SCOP++ and transferred afterwards to an image overlay. These two ways of adding new image overlays are described in sections 9.3.2 and 12.2.7, respectively.

When adding an image overlay, a state-switch *overlayName* will be placed on the main window (at the left margin, in the lower half of the window) with *overlayName* corresponding to the name of the image overlay as selected (maximum 12 letters). See figure 13.1 showing a sample project with two image overlays (an orthophoto *OP4772* and the neighboring orthophotho *OP4768*), and a model overlay *dtm* with the displayed isolines.

When adding an image overlay, a directory with the *overlayName* is searched for in the projects directory. If this directory is not yet created, the user is requested to decide whether to create a new directory. In this directory, the file `overlayName.pyr` is searched. This file serves as a description file, where to find the image data file and (if available) higher pyramid level files. If no such pyramid description file is found, the image data file (or the image pyramid) has to be imported first (cf. section 13.4).

13.2. The state-switch *overlayName*

For each image overlay, a state-switch *overlayName* is placed on the main window , which serves as an icon of the respective image overlay. The lights on it show the display state of the image overlay. Two display states may be selected in the pop-up menu with the left mouse button: *display* or *off* (cf. section 5.1.1 for a general description of state-switches). If the state of the state-switch *overlayName* is changed from *off* to *display*, the image will be displayed. Usually the image will appear immediately. However, sometimes this process can take some time especially if the image is large and no pyramid is available. Another reason for delay can be that more raster graphics files are to be mixed. In the following cases, the user is requested to import a file or is asked for a decision before starting a lengthy calculation (in any of these cases the user can cancel displaying the image):

- No image data file is available. The user will be requested to select a file (the import dialog will be opened, cf. section 13.4).

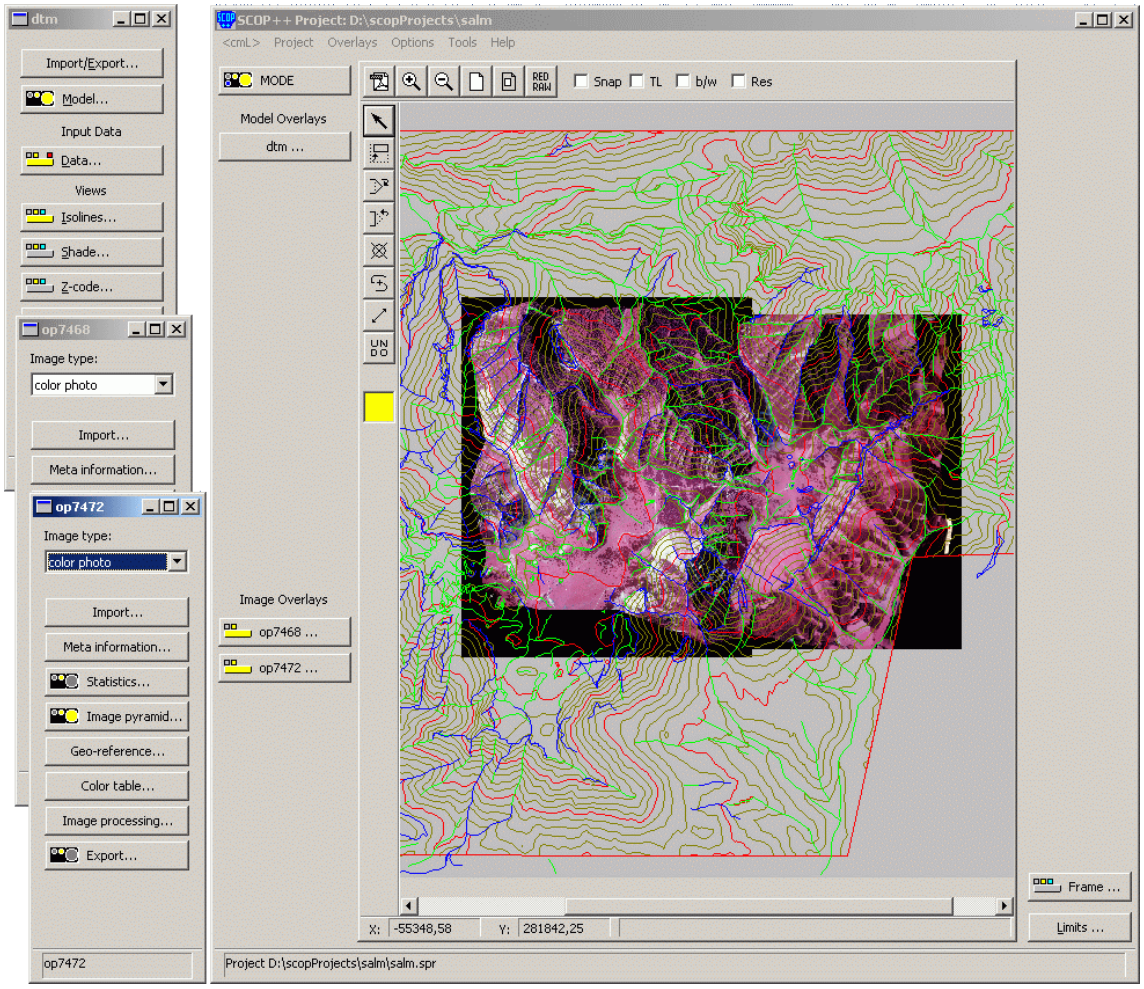
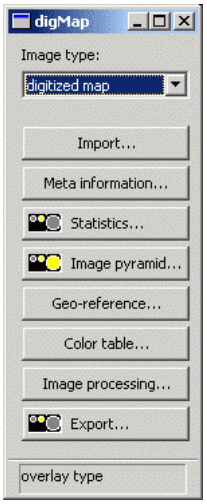


Figure 13.1.: Sample SCOP++ project with two image overlays.

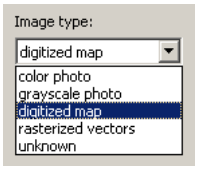
- The image is larger than 9 000 000 pixels (e. g. 3 000 x 3 000 pixels) and no image pyramid is available. In this case, the user is asked whether to calculate a pyramid first (cf. section 13.7).
- The image is to be mixed with another raster graphics file and the result would be larger than 4 000 000 pixels (e. g. 2 000 x 2 000 pixels) when maintaining full resolution. The user is asked, whether he wants to create the mixed image in a reduced resolution. If he decides to accept reduced resolution (*yes*), a higher level of the image pyramid will be taken for mixing or – if there is no pyramid available – the mixed image will have the same size of its pixels as that image of the images to be mixed which has the lower resolution. It may happen that even after reducing resolution, the mixed image will be larger than the threshold. In that case, the operator is asked once again whether he really wants to create the mix.

The window corresponding to the state-switch *overlayName* can be seen at the right side and will be described in the following sections: why to declare the image type (13.3), how to import the image data (13.4), which meta data can be viewed (13.5), how to get a histogram of color/graytone values (13.6), why to derive an image pyramid (13.7), how to set the coordinates of the image in the reference system (13.8), how to change colors and how to set single colors transparent (13.9), which types of image processing operations are available (13.10), and finally how to export the edited image to a new file (13.11).



13.3. The selection *Image type*

If a new image overlay is added, the image type is set to *unknown*. There are several reasons why the operator should specify the type of the image. The available types of image overlays can be seen in the figure beside of this text.



SCOP++ allows to simultaneously display multiple raster graphics. Overlapping images are overlaid and merged to a composite image in the overlap area. Various merging rules are applied (e. g. adding one image to the other or averaging the intensity channel of a color image with the value of a gray-level image). Depending on the selection *Image type* the appropriate merging rule is applied automatically (cf. section 5.3.3)

Image mixing is performed automatically when the user switches more than one raster graphics to *display*. Not all types of images can be merged to meaningful combinations. If no appropriate mixing rule is available, only the image switched on at last will be displayed, and previously displayed ones are switched off. Images of type *unknown* cannot be mixed with any other image.

Currently, the following mixing rules are applied depending on the types of images involved:

- Images of type *color photo* are first transformed from the RGB (red-green-blue) color space to the IHS (intensity-hue-saturation) space. Then, the intensity channel is

modified. Finally the result is transformed back to the RGB space. The rules for modifying the intensity channel are:

  - mixing with another color photo: averaging the intensities.
  - mixing with a gray-scale photo: averaging the intensity value with the gray level.
  - mixing with a digitized map or with rasterized vectors: see below.
  - mixing with another mixed image: only allowed for a gray-scale mixed image (averaging intensity and gray level).
  - mixing with a hill-shaded or gray-scale Z-coded view: shifting the view into a range between -128 and 128 and adding it to the intensity.
  - mixing with a color Z-coded view: the same as with another color photo.

- Images of type *grayscale photo* can usually be mixed with any other kind of image (except of type *unknown*):
  - mixing with a digitized map or with rasterized vectors: see below.
  - mixing with any other type: averaging the gray level with the gray level/intensity value of the other image.
- Images of type *digitized map* or *rasterized vectors* can be displayed together with other images in two ways. They can either be mixed with other images or – if any of its colors is set transparent (cf. section 13.9) – they are displayed covering other raster graphics files:
  1. Displaying multiple images without mixing: This is the case if for the image of type *digitized map* or *rasterized vectors* at least one of its colors (i. e. layers of the map) is set transparent. The map/the vectors are displayed covering other images in those parts of the map which are not set transparent. In the other parts of the image, i. e. behind layers which are set transparent (e. g. the background of an image of rasterized vectors), the other image is displayed. In fig. 13.2, an example of a mix of an elevation-coded with a hill-shaded view of a model behind a digitized map in which all area based layers (except water) are set transparent is shown.
  2. Mixing multiple images: If none of the layers of the map /vectors are set transparent, the underlying image is mixed with the map/vectors by calculating a combined image. The following rules are applied:
     - Mixing of a digital map with a gray-scale images is done by transforming the map into IHS color space, adding the gray-scale image and subtracting 128 from the result. Finally, the result is transformed back to RGB color space. This mixing rule is especially suited for adding a hill-shaded view to a map.
     - Mixing of a digital map with another color image is done by transforming both images to IHS, adding the intensity values of both and transforming back the result to RGB color space.
     - Mixing of rasterized vectors is done by adding the vectors to the other image (either directly to the gray values or to the intensity values after IHS transformation).

13.4. The window *Import image*

When adding a new image overlay, usually no image data file is assigned to the overlay (cf. section 13.1). This has to be done in a second step using the window *Import image* behind the button *Import...*. Importing an image data file can be done even if there was already assigned an image data file to the image overlay (in this case the image will be replaced).

The window *Import image* allows to select an image data file or an image pyramid description file. Several standard file formats of raster graphics as well as proprietary file formats can be interpreted. It is strongly recommended to import geo-coded image data files (i. e. files in which the position and extent of the image in the reference coordinate system is known). If the geo-reference is unknown, it is set to the currently selected views limits (cf. section 4.4) and can be edited afterwards (cf. section 13.8). Some types of im-

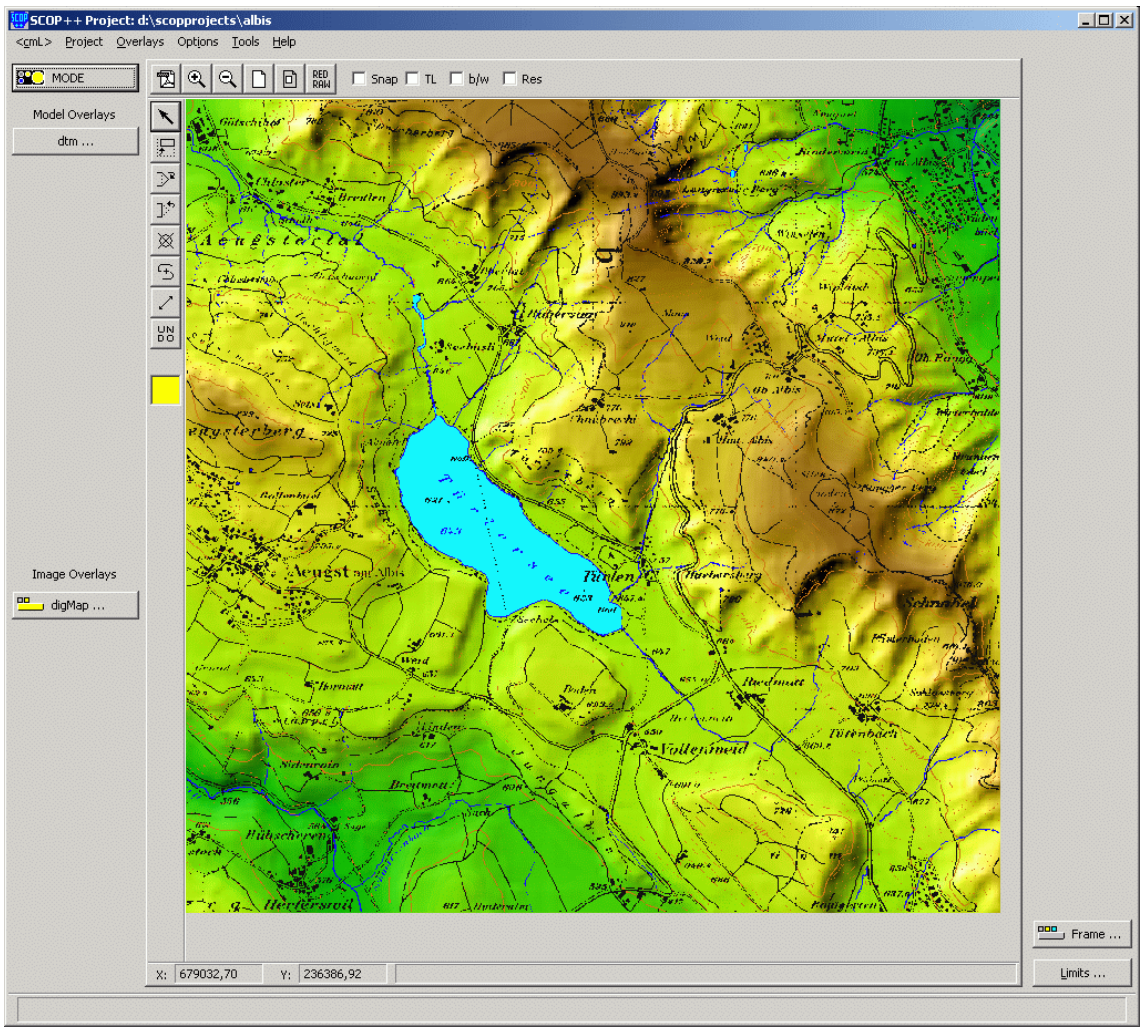


Figure 13.2.: Combined displaying of an elevation coded view, a shading, and a digitized map. Data: (c) Federal Office of Topography, Switzerland, www.swisstopo.ch

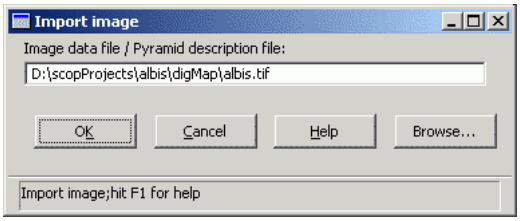


Figure 13.3.: Import of an image data/pramid description file.

age data files have a more sophisticated pixel order in so-called tiles allowing for faster reading of parts of the whole image. The following types of files can be imported:

TIFF: Both tiled and untiled TIFF are supported. Geo-coded TIFF is supported, too.

SCOP PIX: The proprietary image data format of SCOP is tiled and geo-coded.

JPEG: These compressed image data files are not geo-coded.

SCOP pyramid description files: A special type of files which can be imported instead of primary image data files, are "SCOP PYR" (pyramid description) files. These are ASCII description files referring to a series of image data files in different resolutions in any of the listed file formats.

The image data file can be located anywhere on the disk, not necessarily in the image overlay directory. In this image overlay directory, a description file with a link to the original image data files is stored. Note that for some operations the user must have full permissions in the directory of the original image data file(s) (thus selecting an image on a CD-ROM might cause problems):

- When calculating the image pyramid, the image data files of upper levels are written to the directory of the original file (cf. section 13.7).
- Temporary files for image processing (cf. section 13.10) are created in the directory of the original image.
- Defining the geo-reference results by writing a "world" file including the geo-coding data in the directory of the original image.
- Opening SCOP PIX files needs full privileges for the file.

A valid image (image pyramid) is a prerequisite for the following features of SCOP++ concerning image overlays. Thus, as long as no image data file has been selected,

- other buttons of the image overlay are not accessible to the user,
- and the window *Import image* will open, when the user changes the state of the state-switch corresponding to the image overlay to *display*.

Selecting a new image data file (image pyramid) for the image overlay is allowed at any time.

13.5. The window *Meta information*

Clicking on the button *Meta information...* with either the left or the right mouse button opens a window displaying the meta data of the image. The directions of the x and y axis are right and up, respectively.

These data are read from the image data file. In this window , they are not editable.

The data of the first four lines define the transformation between the pixel co-ordinate system of the image and the corresponding user co-ordinate system . This transformation is defined by three of the four data pairs, the fourth is redundant information (cf. figure 13.4).

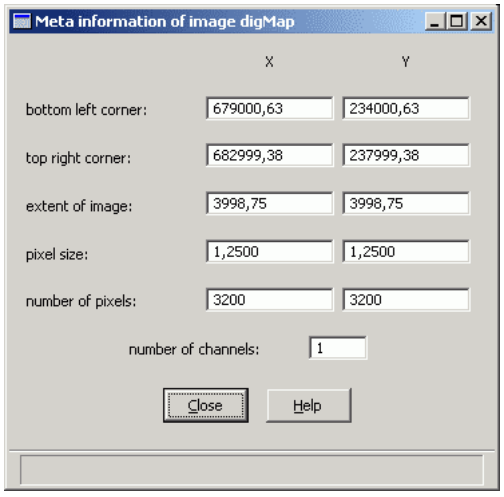


Figure 13.4.: Meta data of image.

- *bottom left corner*: The User Co-ordinates of the centre of the bottom left pixel of the image data file are displayed.
- *top right corner*: The User Co-ordinates of the centre of the upper right pixel of the image data file are displayed.
- *extent of image*: The extent of the image as defined from the centre of the bottom left pixel to the centre of the top right pixel is displayed in units of User Co-ordinates.
- *pixel size*: The size of one pixel in user co-ordinate units is displayed.
- *number of pixels*: The extent of the image in pixel co-ordinate units is displayed.

In the bottom line (*number of channels*), the number of spectral bands of the image is displayed (grey level images have one band, colour images have three bands, some satellite images have even more).

Selecting the button *OK* closes this window.

13.6. The window *Statistics of gray-tone/colour values*

The window showing the statistical data can be opened by clicking state-switch *Statistics...* with either the left or right mouse button.

The button state-switch *Statistics...* has two states: *unknown* or *calculated*. These two states are indicated by small round lamps on the left side of the button (cf. the figure before section 13.3), showing grey or yellow lights, respectively. The current state is indicated by the large round lamp. By default the state *unknown* (gray light) is selected. In order to derive the statistical data the button *Calculate* in the window with the statistical data has to be pressed. The action starts a (sometimes lengthy) analysis of the image data. During the processing time the large lamp of state-switch *Statistics...* is blinking. As soon as the calculation has finished, a (constant) yellow light appears showing that the state *calculated* has been reached, and the button *Calculate* is made inaccessible. Note, that after each change of the image data (e. g. by any image processing routine) the state of state-switch *Statistics...* will be reset to *unknown* automatically.

For monochromatic images the statistical distribution of the grey-tone values is analysed, whereas for colour images the statistics can be calculated in two different modes: *RGB*

(red, green and blue channel) or *IHS* (intensity, hue and saturation of colour). Depending on the selected mode, the displayed state of state-switch *Statistics...* is *unknown* or *calculated*. All combinations are possible: the statistics can be *unknown* in both modes, *calculated* in one of the two modes or *calculated* in both modes. The light is gray or yellow depending on the selected mode.

After calculating the statistics, the number of *used pixels* per band is displayed in the corresponding numerical field. If no border line (see later) was used, this number is equal to the number of pixels of the image, otherwise to the number of pixel centres inside the polygon.

If the statistics of monochromatic images is known, a histogram of the grey level distribution is drawn in the graphics area. For colour images three histograms corresponding to the respective channels red/green/blue or intensity/hue/saturation are drawn. The numerical fields for the following quantities are made accessible for each of the channels:

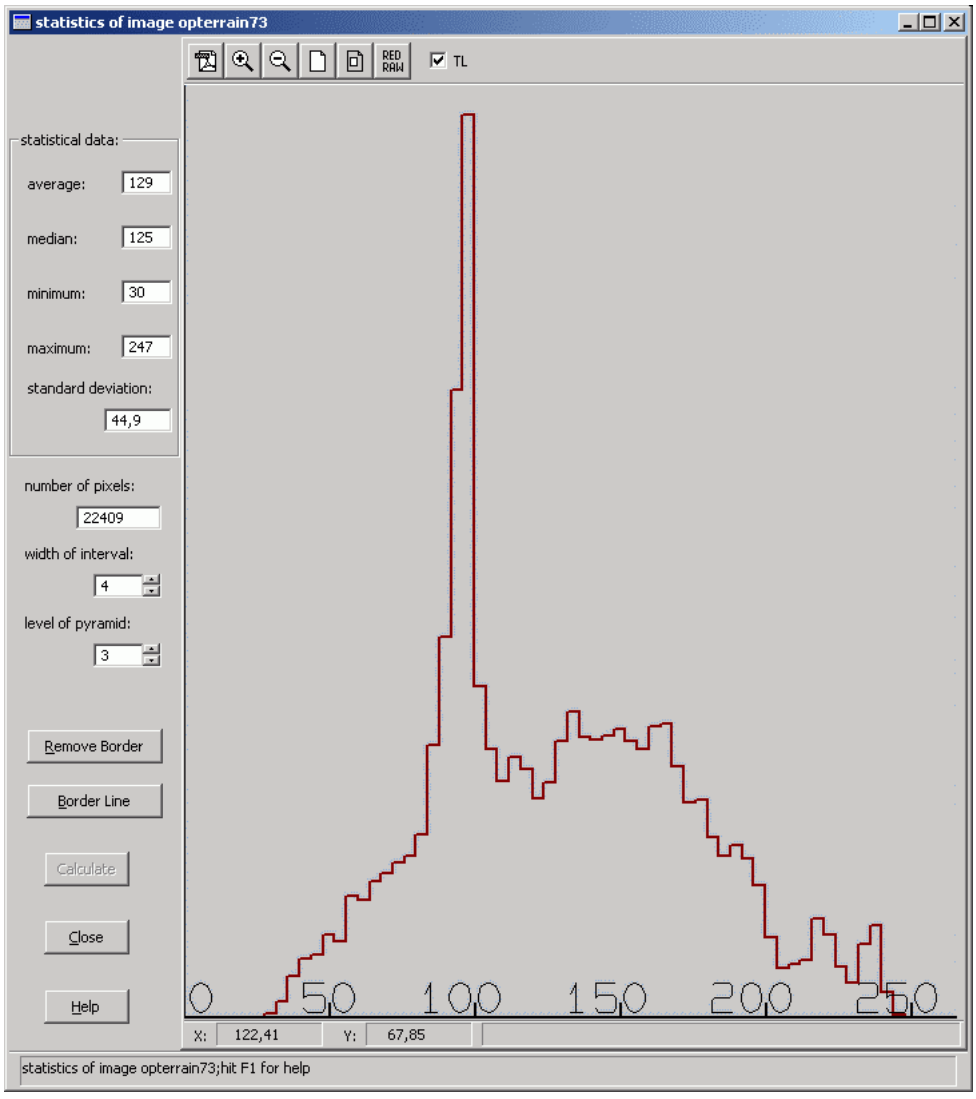


Figure 13.5.: Image statistics.

- *average*: the average value $\bar{g} = \sum \frac{g}{n}$
- *median*: that value for which one half of all values is greater (or equal) and the other half is smaller (or equal)
- *minimum*: the smallest value
- *maximum*: the greatest value
- *standard deviation*: $\sigma = \sqrt{\frac{\sum (g-\bar{g})^2}{n-1}}$ with g being the grey-tone/colour value and n the number of pixels involved.

The *width of interval*, corresponding to the width of the vertical bars in the histogram, is editable. The displayed co-ordinates at the bottom of the graphics area show the position of the mouse in the graphics. The x co-ordinate is scaled to the grey-tone/colour values (in the range from 0 to 255), whereas the y co-ordinate is scaled to fit the whole histogram best into the graphics area. The buttons on the top of the graphics area have the usual function as explained in section 8.

If the image to be analysed is represented as image pyramid, the statistics can be calculated in any of the *pyramid level*s. By default, the highest level is selected in order to make the calculation fast. Nevertheless, the statistics of the highest pyramid level may differ slightly from the statistics calculated from the original image (lowest level, i.e. the level 0). For most purposes these differences can be ignored.

As some image processing algorithms need information about the distribution of grey-tone values (of monochromatic images) or of the intensity channel (of colour images), the calculation of the statistics may be started by the system (e.g. image enhancement of colour images by histogram normalisation requires statistics in *IHS*). In this case, again the lamp on state-switch *Statistics...* will blink while analysing image data.

If the user is interested in the statistics of a specific part of the image, an arbitrary closed polygon can be introduced as border line. This functionality is not included in "SCOP++ Kernel". Therefore the button *Border Line* and the button *Remove Border* are locked. The statistics window can be closed by the button *Close*.

13.7. The window *Pyramid of image overlayName*

High resolution images are stored in huge image data files. Reading these files takes some time. Even for displaying the image as overview in a lower resolution, the entire file has to be read. We recommend to create separate lower resolution files in advance in order to speed up the display.

These image data files of lower resolution are stored in a so-called image pyramid. The lowest level of the pyramid corresponds to the given image in the original resolution. Each higher pyramid level contains an image representing the whole image content in a lower resolution. The single pixel of the higher level covers a larger area while the number of pixels is reduced.

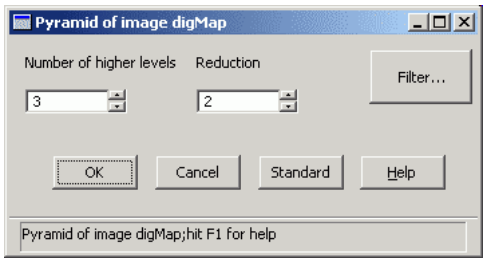


Figure 13.6.: Image pyramid parameters.

If the image overlay already has an image pyramid (e. g. a SCOP pyramid description file was imported, cf. section 13.4), the light on the button *Image Pyramid...* will be yellow, otherwise gray. The parameters of the pyramid can be viewed in the window *Pyramid of image overlayName* which can be opened by clicking at the button *Image pyramid...* (cf. fig. 13.6). The numeric field *Number of higher levels* shows, how many higher level images are available in the current image pyramid in which the original image data file is not counted. Further, the reduction factor and filter matrix which were used to create these pyramid levels can be seen. If the pyramid is not yet created (i. e. the numeric field *Number of higher levels* is zero), the window *Pyramid of image overlayName* can be used to prepare the parameters. For large images, it is strongly recommended to let a pyramid be created, because displaying the image (and especially zooming in and out) will be significantly faster.

The reduction factor of the image pyramid can be selected in the numeric field *Reduction*. The value given there is interpreted as one-dimensional reduction factor, i. e. as the relation of the number of pixels of one image line (or one image column, respectively) between two consecutive pyramid levels. On the other hand, the reduction factor can be interpreted as number of pixels of the lower level image (in one dimension of the image), which are covered by the next higher level image. The total number of pixels is reduced by a factor of the square of the value specified in the numeric field *Reduction*. For example: Given a reduction factor of 2, an image of extent 2.000 x 2.000 pixels (i. e. 4 MB) is reduced to an image of 1.000 x 1.000 pixels (i. e. 1 MB). The next higher level would have 250 kB.

The button *Standard* will prepare all parameters for deriving an appropriate image pyramid. The number of higher levels to be created is set to three. Depending on the type of color representation of the image (cf. section 13.9 for a description of these types), the standard filter is set to

- a 3 x 3 binomial filter (color or gray-scale images),
- a 3 x 3 median filter (palette images).

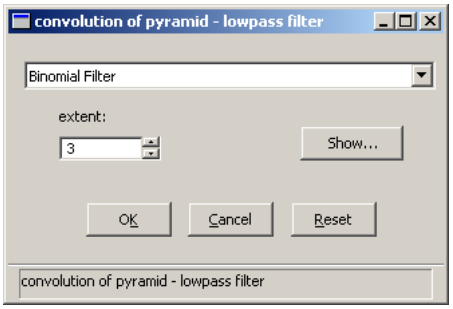


Figure 13.7.: Selection of a convolution filter for pyramid calculation.

Both filters are lowpass filters. The binomial filter is similar to a Gaussian smoothing filter (but can be processed more efficiently, because it is separable in the two directions). The median filter selects one of the existing gray-level values (or color numbers) instead of calculating an average value. It selects the one for which one half of the values in the neighborhood is smaller and the other half is larger than itself. In the window *convolution of pyramid - lowpass filter* behind the button *Filter...* (cf. fig. 13.7), other filters can be chosen and the extent of the filter can be changed.

The filter matrix or a verbose filter description can be viewed in the window *filter for pyramid - lowpass filter* behind the button *Show...* (cf. fig. 13.8. Read more about convolution filters in section 13.10.4).

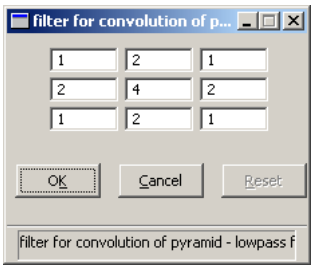


Figure 13.8.: Binomial filter.

If the parameters have been changed, the calculation of the respective pyramid level files is started after closing the main pyramid definition window *Pyramid of image overlayName* by the button *OK*. Closing the window by the button *Cancel* will reset the parameters and won't start the calculation.

The process of deriving all higher level files may take some time. During this time the button *Image Pyramid...* will blink, the state-switch *overlayName...* will be made inaccessible and the process is registered as background process (cf. 5.3.12). As soon as calculation has finished, the state-switch *overlayName...* is made accessible again.

Higher level pyramid files are created in the same directory as the original image data file. Thus, writing permissions must be set for that directory (note that calculating the pyramid for an image on a CD-ROM is not possible). The names of the higher level files are the same as the original file plus "`_ii`" before the extension (with `ii` starting at `01` representing the level number). A file with extension `*.pyr` in the directory of the original file serves as description file how many and which sub-levels are available.

13.8. The window *Geo-reference of overlayName*

The geo-reference describes the position and extent of the image according to the reference coordinate system. The following types of geo-reference are supported:

- world files: The geo-coding is read from separate files (e. g. tiff-world files `*.tfw` or jpeg-world files `*.jgw`). These files are only accepted if they are available obeying the naming convention rules: The name of the world file must be the same as the name of the image data file and must be located in the same directory. The extension of the world file is created from the extension of the image data file from its first and last letter plus "w" (e. g. the world file for an image data file `c:\x.tif` is `c:\x.tfw`). If the extension of the image data file has more than three letters, the letter "w" is directly appended to that extension.
- geo-reference written on file: See section 13.4 to read which types of image data files may contain the geo-reference.

For images whose geo-reference is unknown, the operator can provide the data interactively by selecting the button *Geo-Reference...* and opening the window *Geo-reference of overlayName* (cf. fig. 13.9).

As long as the window *i*s open, the current position and extent of the image is drawn as red rectangle in the main graphics panel. In the parameter window, it is also symbolized by a rectangle. The coordinates of the corners of the image can be seen in the numeric fields *X1, Y1, X2, Y2*. When importing an image data file whose geo-reference is not known, its placed inside the current views limits (cf. section 4.4). In that case, the numeric fields *X1, Y1, X2, Y2* are available for editing:

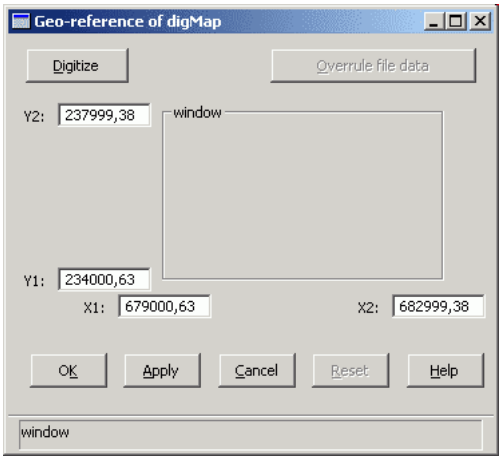


Figure 13.9.: Geo-coding parameters.

- By editing the coordinates of the lower left corner (the numeric fields *X1, Y1*) the image is moved, i. e. the coordinates of the upper right corner (the numeric fields *X2, Y2*) are changed, too.
- By editing the coordinates of the upper right corner (the numeric fields *X2, Y2*) the image is stretched, i. e. the coordinates of the lower left corner (the numeric fields *X1, Y1*) remain unchanged.
- Pressing the button *Digitize* allows digitizing the window at the main graphics panel.

After accepting the new geo-coding by selecting the button *OK* or the button *Apply*, a "world file" containing the new geo-reference is written in the directory of the original image data file. (Note, that writing permissions must be available in that directory).

If an image data file is imported, for which the geo-coding is already available, all editing elements are locked. The current geo-coding can be viewed but not edited. In order to unlock these elements, the button *Overrule file data* has to be pressed. In this way, the current geo-reference is overridden. Already existing "world files" are overwritten. Note, that overruling the image file's geo-reference is not possible for images of type "SCOP PIX".

For writing a command file procedure, be aware of the rule that first the image must be moved (by setting the values of the bottom left corner, i. e. the numeric fields *X1, Y1*), and afterwards the image can be stretched (by setting one of the values of the upper right corner, i. e. one of the numeric fields *X2, Y2* – the second one will be updated automatically):

cmL example

```
@setGeoCoding;
//Image Overlays
OP4772/
  Geo-reference/
    Overrule,
    X1 = 57800.000,
    Y1 = 5348000.000,
    X2 = 58300.000;
@endProc;
```

13.9. The window *Palette of image overlayName*

Concerning the color representation, digital images can be divided into several types:

- Gray-level images: For each pixel, the gray value in the range between 0 and 255 is stored. The storage amount is 1 byte per pixel.
- Palette images: For each pixel, the color number is stored. 256 colors are allowed so that the color number can be coded with 8 bits. The conversion between the color number and the color is stored as so-called look-up table. Again, the storage amount is 1 byte per pixel.
- Color images (true color images): For each pixel, three color values in the channels red, green and blue are stored. The storage amount is 3 bytes per pixel.
- Other types (e. g. binary images or 16-bit color images).

For the second group of images, the palette (look-up table) is stored on the image data file providing the three color values (red, green and blue) for a color number. If the image of an image overlay is of type palette image, the button *Color Table...* will be accessible. The window *Palette of image overlayName* is designed to display the available colors (cf. fig. 13.10), to select some of these to set them transparent, and (or), to exchange single colors of the palette..

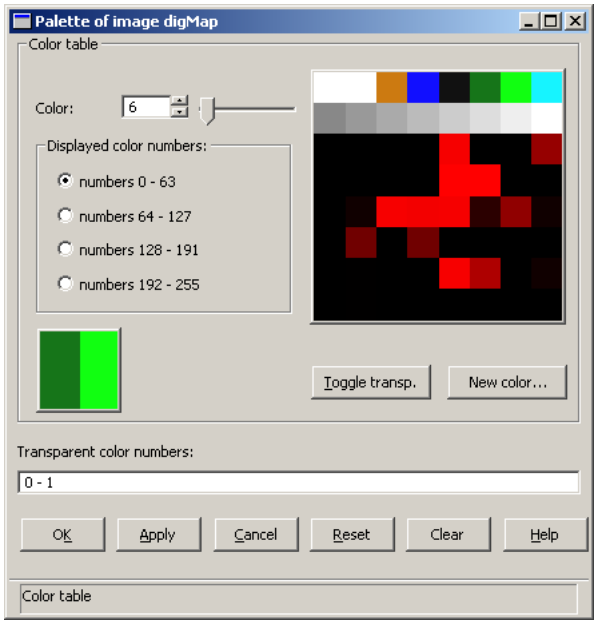


Figure 13.10.: The color table of a ”palette image”.

Transparent colors: For some images (e. g. digitized thematic maps), it may be of interest to set one (or more) color(s) transparent. That means, that pixels with the specified color number are not drawn. The graphics content is thus not changed at those positions, i. e. underlying graphics elements remain visible. For the example of a thematic map, the background and some thematic layers may be set transparent, while other layers should be displayed. In this way, underlying other digital images (e. g. an orthophoto) can still

be seen, even if some thematic layers are overlaid. In section 4.7 an example on how to use transparency was described (the sample project "albis"). See section 13.3 for reading how to suppress mixing of raster graphics and just overlaying a map.

In order to select a color number for setting it transparent (or for removing it from the list of transparent color numbers), the color is first chosen in the group *Color table*. In the bottom left color field, the current color and the previously selected color are drawn (right and left half, resp.). The color number can be chosen by

- typing the number in the numeric field *Color*,
- dragging the slider right of it,
- selecting a range of color numbers in the group *Displayed color numbers* to be displayed right and clicking at any of these colors.

Accepting a color by clicking on the button *Toggle transp.* either adds this color number to the text field *transparent color numbers* or (if it is already set transparent) removes it from this list of transparent color numbers.

In text field *transparent color numbers*, a comma-separated list of color numbers is displayed. This list is automatically updated each time the user selects a color number in the group *color table*. Nevertheless, it can directly be edited by the user, too. Note that a comma (",") has to be typed in to separate color numbers.

Editing the palette: Each color of the palette can be exchanged separately by another color. First, a color number has to be selected (see the list above how this can be done). After pressing the button *New color...*, a window for selecting the new color is opened (cf. section 12.8.6 for a description on how to select a color).

Accepting the changes: In order to accept the list of transparent colors and (or) the edited palette and to forward it to the main graphics panel, the button *Apply* or the button *OK* (additionally closes the window *Palette of image overlayName*) has to be chosen. The button *Reset* resets changes. Selecting the button *Cancel* performs resetting and closes the window . The button *Clear* clears the list of transparent colors and resets the palette to that which is written on the image data file.

In order to write the edited palette on the image data file, the image overlay has to be exported to a new file (cf. section 13.11).

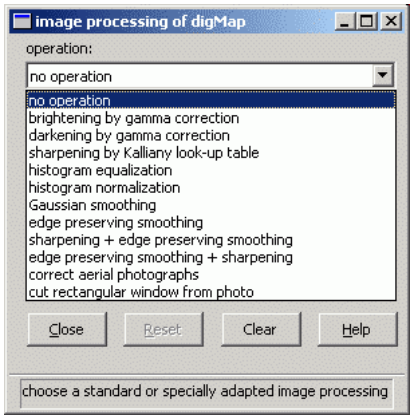
13.10. The window *image processing of overlayName*

For image overlays, some image processing operations are available. The window *image processing of overlayName* can be opened by clicking at the button *Image Processing...* with either the left or the right mouse button.

For understanding the following description, it is important to learn the terms used: Any image processing OPERATION can consist of one or more image processing STEPS. These steps can have different types that will be explained later.

The window *image processing* needs at first selecting an *operation* in the first line. Several operations may appear in this selection. First, a number of standard operations is offered. In "SCOP++ Kernel", no further operations are available. In "SCOP++ Visualizer"or in ORPHEUS, the operator can define special operations and save them to file so that they can be used in other projects, too.

The predefined standard operations are:



- *no operation* is always offered as the first "operation" and it is selected by default.
- *brightening by gamma correction* is an operation consisting of one image enhancement step with a global contrast enhancement by a gamma correction with $\gamma = 0.7$ (cf. section 13.10.3).
- *darkening by gamma correction* is the opposite operation with $\gamma = 1.3$ (cf. section 13.10.3).
- *sharpening by Kalliany look-up table* has one step: a local contrast enhancement by a Kalliany look-up table (cf. section 13.10.3).
- *histogram equalization* is a contrast enhancement operation which tries to reach horizontal line in the histogram of color intensities (gray values) (cf. section 13.10.3).
- *histogram normalization* is a contrast enhancement operation which tries to reach a Gaussian distribution of color intensities (gray values) (cf. section 13.10.3).
- *Gaussian smoothing* is a convolution with a Gaussian smoothing filter (cf. section 13.10.4).
- *edge preserving smoothing* is a convolution with a sigma filter (cf. section 13.10.4).
- *sharpening + edge preserving smoothing* and *edge preserving smoothing + sharpening* consist of the two steps: local contrast enhancement by a Kalliany look-up table (cf. section 13.10.3) and the convolution by a sigma filter (cf. section 13.10.4).
- *correct aerial photographs* has one step of correcting the radial intensity fall-off (cf. section 13.10.5).
- *cut rectangular window from image* is one step of cutting the image (cf. section 13.10.6).

Selecting one of these standard operations lets the appropriate buttons (called state switches) be placed in the lines *1st step, 2nd step, ...* These state switches are labeled with the digit of the processing order and the type of the processing step (e.g. figure 13.11: *1 - Convolution...* and *2 - Contrast...*).

For each of these state switches a window can be opened for specifying the parameters of the respective image processing step. This parameter window is opened either by clicking at the state switch with the right mouse button or by selecting *Properties...* from the pop-up menu with the left mouse button (cf. fig. 13.12). Note, that for standard operations the parameters of the image processing steps are already set to default values and usually need not to be adapted. Changing parameters of standard operations is not part of the functionality of "SCOP++ Kernel", but only of "SCOP++ Visualizer"or ORPHEUS.

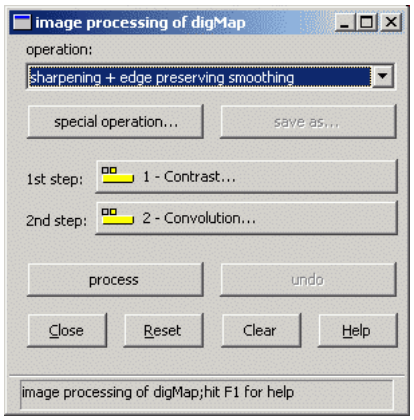


Figure 13.11.: Two state switches indicating an operation consisting of two steps.

Additionally, the image processing buttons have two states: *deactivated* and *active*. The two possible states are indicated by two small quadratic lights in the top left corner of the button (grey and yellow for preview off and on, respectively). The selected state is indicated by the colour of the larger bar below. By default the second state (*active*) is selected. The states are changed in the pop-up menu (left mouse button).

The calculation of the entire image (pyramid) is started by pressing the button *process* in the last line of the box *image processing*. If any of the image processing state switches is not set to state *active*, the user will be asked, whether this step should be used for processing the image or not.

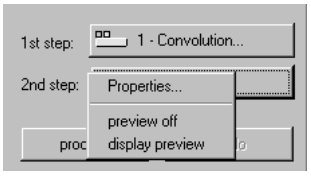


Figure 13.12.: Pop-up menu of image processing state switches.

Depending on the size of the image the operation can require a certain amount of time. During this time the light on the respective state switch blinks, the button *undo* right of the button *process* remains inaccessible and no image can be seen on the display. Furthermore, the process is registered as background process which can be interrupted (cf. section 5.3.12).As soon as the calculation of the new image (or the new image pyramid) is finished, it is displayed and the button *undo* is made accessible.

Image processing operations produce a new image data file. The new file will have the same name as the original one and will receive the extension `*.tp0` or `*.tp1`. It will be located in the image overlay directory. If the new image data file has successfully been created, the pyramid description file in the image overlay directory `ovlName.pyr` is updated so that it points to the new image data file. The original image data file is not changed. If the old image data file was the ground level of an image pyramid, the pyramid is rebuilt.

If the resulting image is not satisfying, further image processing operations can be invoked or the last one can be reversed. The latter is done by pressing the button *undo* and thus updating the pyramid description file to point to the original file again. After that the button *undo* will be locked, because only one image processing step can be undone!

As mentioned at the beginning of this section, four possible types of image processing steps are available and are used in the standard operations:

- enhancing the contrast of the image (cf. section 13.10.3)
- applying a convolution with a filter (cf. section 13.10.4)
- correcting the intensity fall-off in aerial photographs (cf. section 13.10.5)
- cutting a rectangular window from the image (cf. section 13.10.6)

These operations may be edited and adapted for a special application, or, even a new application may be created, which will be explained in the following section.

13.10.1. Special image processing operations

The button *special operation...* is responsible for defining application specific image processing operations. This functionality is not included in "SCOP++ Kernel", but only in "SCOP++ Visualizer"or in ORPHEUS.

13.10.2. Save operation to file

The functionality behind the button *save as...* is not included in "SCOP++ Kernel", but only in "SCOP++ Visualizer"or in ORPHEUS.

13.10.3. Contrast Enhancement

The parameters of contrast enhancement steps can be defined in the window behind the respective state-switch *Contrast....* In "SCOP++ Kernel", only standard image processing operations without parameter modification are available. Full functionality is offered in "SCOP++ Visualizer"or in ORPHEUS.

13.10.4. Convolution

The parameters of convolution steps can be defined in the window behind the respective state-switch *Convolution....* In "SCOP++ Kernel", only standard image processing operations without parameter modification are available. Full functionality is offered in "SCOP++ Visualizer"or in ORPHEUS.

13.10.5. Correction of intensity fall-off in aerial photographs

The parameters of steps correcting intensity fall-off in aerial photographs can be defined in the window behind the respective state-switch *Fall-off....* In "SCOP++ Kernel", only standard image processing operations without parameter modification are available. Full functionality is offered in "SCOP++ Visualizer"or in ORPHEUS.

13.10.6. *Cut*

Selecting state switches of type *Cut...* on the Image Editor window with the right mouse button or selecting *Properties...* in the pop-up menu with the left mouse button opens the parameter window for preparing cutting a rectangular region of interest from the image (figure 13.13). This parameter window can be closed without cutting the image by pressing either the button *OK* or the button *Cancel*. *OK* accepts the selected parameters, *Cancel* resets them. *Reset* becomes available after changing any of the parameters. *Clear* becomes available, if a window is defined that is smaller than the domain of the image. Selecting *Clear* will reset the window to the image domain. See section 13.10 how to start processing the image.

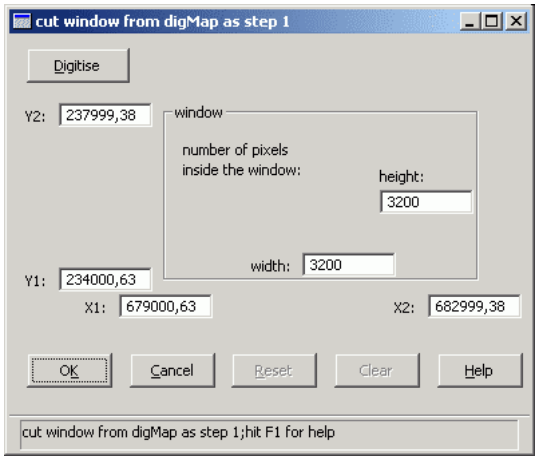


Figure 13.13.: Parameter window of cutting steps.

The window of the new image is to be given in User Co-ordinates by two points, the lower left and the upper right corner. The new image will contain all pixels that are at least partially inside the window.

If the co-ordinates of the window are known, they can be typed into the numerical fields. Otherwise the button *Digitise* can be pressed (left mouse button) and the window can be digitised in the main graphics panel (by digitising any two opposite corners of the window with the left mouse button). If the user wants to repeat digitising the window, he has to press the button *Digitise* once more. The co-ordinates of the digitised window are written in the respective numerical fields. In these fields only values inside the image domain are allowed.

Only the X and Y co-ordinates of the corners of the window of interest can be changed in the numerical fields. The number of pixels of the resulting image serves as (redundant) information to the user and cannot be edited.

As long as this dialog is on the screen, the window is drawn to the main graphics panel as a red rectangle.

13.11. The window *Export of image overlayName*

Exporting the image to a new image data file may be of interest in various situations:

- For converting the image to another file format.
- After changing the image by an image processing operation (cf. section 13.10).
- For writing a new geo-reference or a new palette to the image (cf. sections 13.8 or 13.9).

In the window *Export of image overlayName* (cf. fig. 13.14), the file name and the file format have to be specified for the export. See section 13.4 for a list of image data formats which are supported by SCOP++. By selecting the button *Browse...*, the file name can be chosen in a file browser.

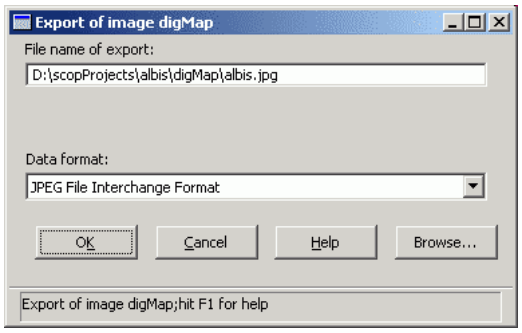


Figure 13.14.: Definition of file name and type for export.

Please note that an appropriate file extension will be appended if the file name has been entered without extension (ie. .jpg for JPEG files, etc.). The file extension will be adapted automatically in response to a change of the data format. This automatism is applied as long as no file extension is specified by the user. Note also that specifying a file type in the the file browser does not influence the setting of the data format.

The button *OK* starts a process in background which will copy (or convert) the current image data file to the specified one. During this process, the lights on the button *Export...* (cf. the figure before section 13.3) will blink so that the operator can see that the export is still working. As soon as the process is finished, the lights will stop blinking. Pressing the button *Cancel* will close the window without starting the export process.

14. Perspective Views

14.1. Concepts

The idea is to get a view of a DTM similar to making an image with a camera. This virtual camera can be placed anywhere in our reference coordinate system and we can look in some direction. These specifications are called *exterior orientation*. Furthermore the camera has some characteristics like image format, focal length, and – if we think of it as a digital camera – pixel size. These specifications are called *interior orientation*. A camera is usually a device using central projection to get an image. The virtual camera used there may also produce other *types of views* like parallel projections or cylindrical projections on a vertical cylinder to get panorama images. Some other parameters may be specified too. Details will be given later.

On the other side we can determine how we would like to see the DTM. The simplest view is to look at the DTM as grid of lines. Every grid point of the DTM is connected with its neighbouring points with lines. Other types of lines included within the DTM (breaklines, formlines, borderlines) can be visualized too. If we have some georeferenced image we can use this image as texture and look at the DTM as a textured model. Images may come from a model overlay like a shading or a z-coding of the DTM or they may come from an image overlay like an orthophoto or a digital map.

The calculations to get the final image may take some time, therefore two modes are provided: a *preview mode* showing only a reduced grid of the DTM to quickly check the changes of orientation parameters, and a *final mode* to get the image with full details according to all the settings we can make.

14.2. Perspective Window

If we add a perspective view to our project using drop-down menu *Perspectives/Add perspective view...* a button with the name of the view will be placed at the right side of the main graphics panel below the label *Perspective Views*. Clicking at this button with the left or right mouse button will open a window like in figure 14.1.

The window consists of three parts. At the upper side there are fields and buttons to specify the orientation parameters, camera characteristics, and contents of the image. The graphics area in the middle of the window shows the reduced grid of the DTM while in preview mode and the final image when in final mode. At the lower edge some buttons are provided to specify an action.

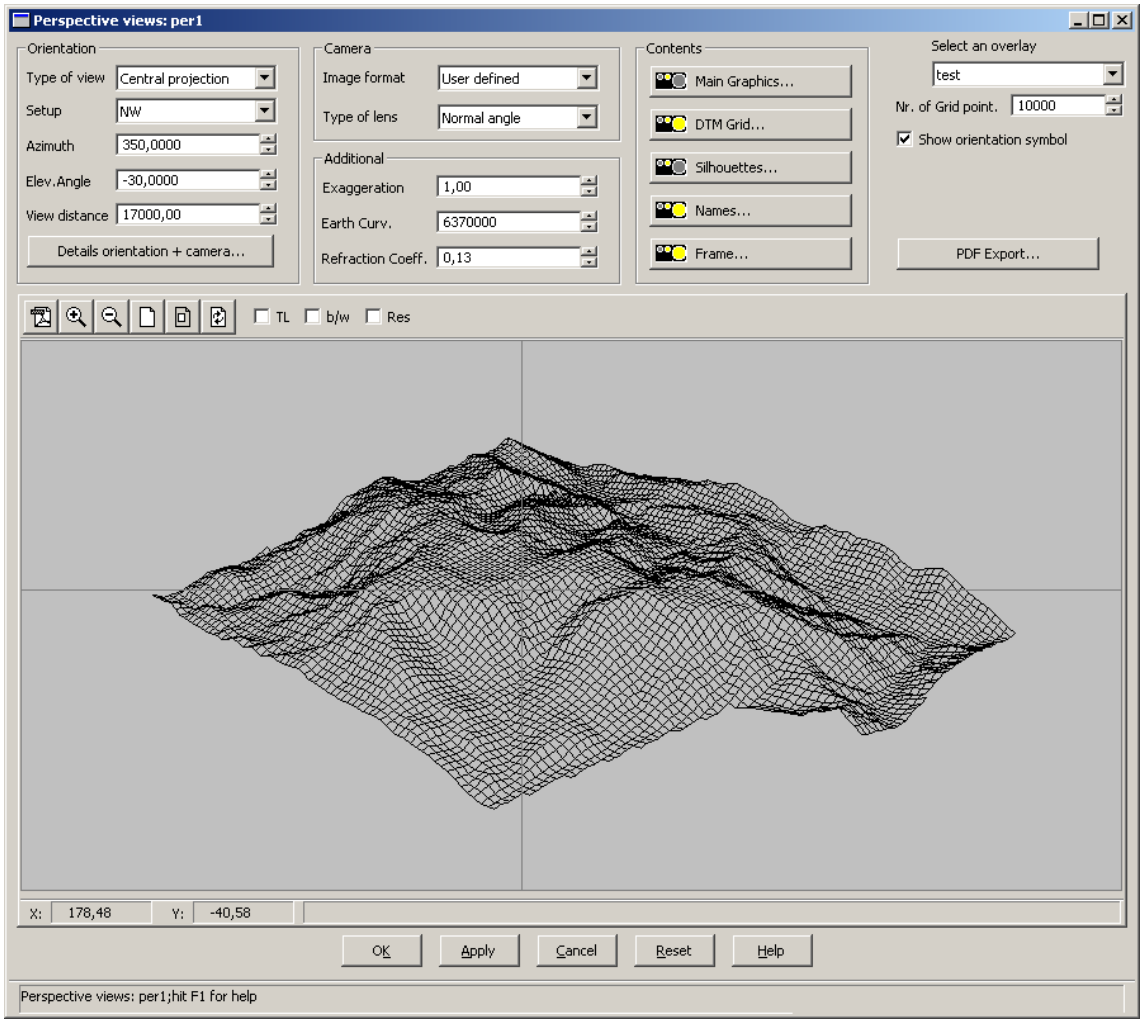


Figure 14.1.: The window *Perspective views: ...*

14.2.1. Selecting an Overlay

A SCOP++ project may contain more than one model overlay. If we want to get a perspective view of a DTM we have to select exactly one model overlay containing a DTM to be used for that purpose. When opening a window *Perspective views: ...* the first time no overlay is selected. The first step is to select a model overlay from the selection *Select an overlay* located at the upper right corner of the window.

14.2.2. Preview Mode

Most of the time the window is in preview mode. This means only a reduced grid of the DTM is displayed in the graphics area and no hidden line removal is done. If some parameters are changed by the user the graphics area will be updated immediately. The density of the grid may be controlled by the numeric field *Nr. of Grid points*. But the given value is only a hint for the program.

14.2.3. Final Mode

Whereas preview mode only provides a limited quickview of the DTM in final mode an image will be made using all the settings we have specified. This may take some time if the DTM is large or large images are to be used. Clicking at button *Apply* will start the calculations for final mode and display the final image instead of the preview. Changing any parameter will automatically switch back to preview mode.

14.2.4. PDF Export

The final image may be exported as file in PDF format. This can be accomplished by clicking at the PDF button in the toolbar of the graphics window or the button *PDF Export...*

14.3. Orientation

The first step to determine a *perspective view* of a DTM is to define the interior and exterior orientation of a virtual camera. The imaging process is mathematically formulated by a perspective transformation which gives the relation between a point in object space and the corresponding point in the photograph (described by *image co-ordinates*) Two coordinate systems are involved with the imaging process:

- Interior orientation: The values are given in units of the x-y-z image coordinate system. These parameters define the characteristics of the virtual camera (focal length, principal point, image size, pixel size).
- Exterior orientation: The values are given in units of the X-Y-Z reference coordinate system. This coordinate system usually corresponds with the national geodetic coordinate system and serves to define the position and rotations of a the camera.

Both coordinate systems are mathematically positive systems (i.e. X-Y-Z values correspond to easting - northing - height).

14.3.1. Interior Orientation (Camera Parameters)

The image format together with the type of lens specification define a simplified inner orientation. These parameters are accessible in the main perspective window by the selection boxes: *Image format* and *Type of lens*. A more detailed specification of the inner orientation is possible in the dialog window *Details orientation + camera*. There are numeric fields within the groups *Inner orientation* and *Image size*. Parameter set *InnOri* specifies the position of the projection center in the image coordinate system, i.e. the first an second value define the position of the principal point (x_0, y_0) and the third value corresponds to the focal length (c ... camera constant). The principal point usually lies in the center of the picture. The description of the image format is made by the numeric fields within group *Image size*. The left lower edge and the right top margin are given in units of the image coordinate system.

14.3.2. Exterior Orientation

For determination of the exterior orientation the following elements are provided:

Type of view: This selection allows to choose between *central projection*, *parallel projection* and *panorama*. Panorama view is defined with a vertical cylinder axis. In this case the focal length of the camera (taken from the inner orientation) corresponds to the radius of the projection cylinder.

Setup: This selection allows to select some predefined directions to look at the DTM:

- North,
- NE - northeast,
- East,
- SE - southeast,
- South,
- SW - southwest,
- West,
- NW - northwest.

In case of such a predefined view the elevation angle is always set to -30 gon (looking downwards at the DTM) for central and parallel projection, and set to 0 gon (looking horizontal) for panorama view. the angles of rotation (azimuth-zenith distance) can be exactly specified my means of the appropriate numeric fields.

***Azimuth, Elev. Angle* and *View distance/Image scale*:** Those numeric fields serve to specify angles, that describe the twist between the camera and the object coordinate system. The angles are given in gon. The complete circle corresponds thereby to 400 gon. The elevation angle is fixed to 0 gon when *panorama* is selected as *type of view*. View distance means the distanc between the projection center (position of the camera) and the target point. The numeric field *Image scale* replaces the numeric field *View distance* in case of parallel projection. We may think of parallel projection as a special case of perspective projection with infinite focal lenth. So we need the image scale parameter to allow a mapping from the large DTM to the usually small image.

***Details orientation + camera*:** This sub-dialog allows to specify exactly the orientation parameter, because all of them are available in numeric form. The selection *Type of rotation* offers additional angle definitions:

- *Terrestrial 1 (Azimuth-Elevation-Kappa)*,
- *Terrestrial 2 (Alpha-Zeta-Kappa)*,
- *Aerial 1 (Roll-Pitch-Yaw)*,
- *Aerial 2 (Omega-Phi-Kappa)*, and

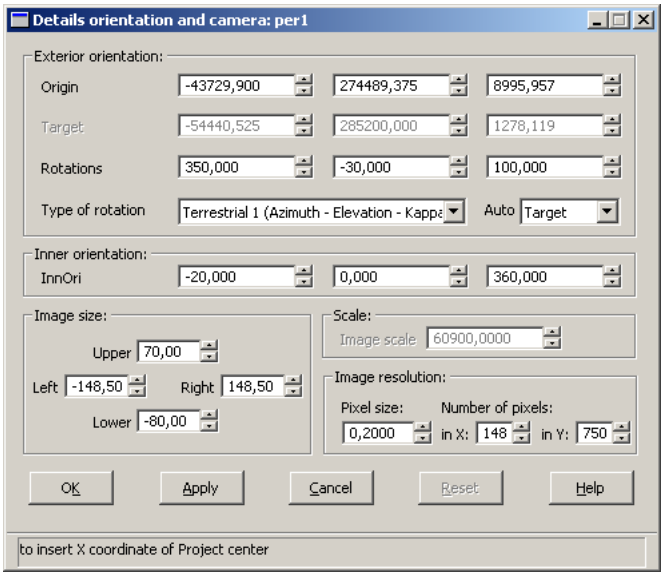


Figure 14.2.: The window *Details orientation and camera: ...*

- *Aerial 3 (Phi-Omega-Kappa)*

complementary to the angle of rotation, defined in the main perspective window in accordance with the terrestrial case (azimuth-elevation-Kappa). The dialog window *Details orientation + camera* offers additional possibilities for an exact definition of the parameters of exterior orientation (projection center, target point and rotation angles). But not every combination of these parameters is permitted. Specifying two of the parameter sets *Origin*, *Target*, and *Rotations* the third of them can be calculated automatically. Therefore we have to choose one of the three parameter sets for automatic calculation using the selection *Auto*. When changing orientation parameters in the main perspective window the corresponding individual redundant parameters in the detail sub-window will be adapted automatically and vice versa, e.g. if the projection center- and/or target-coordinates are changed, the rotation angles are computed automatically and the actual values will be displayed.

wind rose on main graphics panel: The SCOP++ main graphics panel may show some representation of the selected model (e.g. elevation coding or shading). A wind rose will be drawn there at the target point if the checkbox *Show orientation symbol* is selected in the perspective window. The viewing direction (azimuth and elevation angle) will be displayed with help of a yellow line ending at the target point. The direction of the line corresponds to azimuth angle and the length of it to the elevation angle. If the line starts at the outer circle this corresponds to an elevation angle of zero. So the outer circle represents the horizon. When the starting point lies inside the wind rose the elevation angle is negative (looking downwards), when it lies outside the elevation angle is positive (looking upwards).

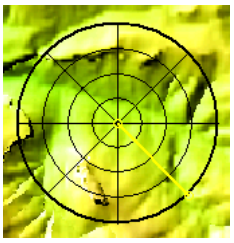


Figure 14.3.: The *wind rose.*

14.4. Contents of Image

The contents of perspective views is based upon the products of the SCOP++ main graphics panel (views, image overlays). Thus a complex definition of contents perspective view is economized. Additionally it is possible to create DTM grid, to draw silhouettes, to annotate locations with names and to draw a frame with title around the image. Using the buttons from the group of *Contents*, elements of the perspective view may be switched on and off. Left-clicking the state-switches displays a pop-up menu *Properties..., Deactivate, Activate*. Left-clicking the item *Properties...* opens a dialog window , where the window contents can be configured. Right-clicking the state-switch is identical to left-clicking the *Properties...* item in its pop-up menu . The other items represent different „states“: left-clicking any of them changes the color of the light on the state-switch, indicating the state it has been switched to.

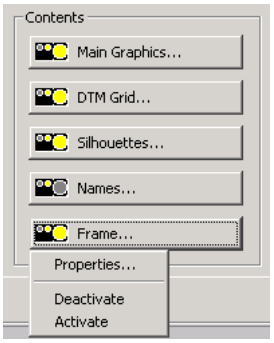


Figure 14.4.: Buttons from the group of *Contents . . .*

14.4.1. Contents from Main Graphics Panel

The *Main graphics...* state-switch displays only two items: *Deactivate, Activate*. Left-clicking the item *Activate* inserts contents from main graphics panel into perspective image and vice versa left-clicking the item *Deactivate* excludes contents displayed within the main graphics panel from perspective image.

Remark: When using parallel projection as type of view then contents of the MainGraphics cannot be displayed in the perspective image. They will be ignored.

14.4.2. Contents from the DTM

The button *DTM Grid...* switches the raster lines on (when item *Activate* is selected) and off (when item *Deactivate* is selected). Left-clicking the item *(*Properties...) opens the dialog window *Details DTM Grid*. If the checkboxes *Grid, Breaklines, Formlines and Borderlines*

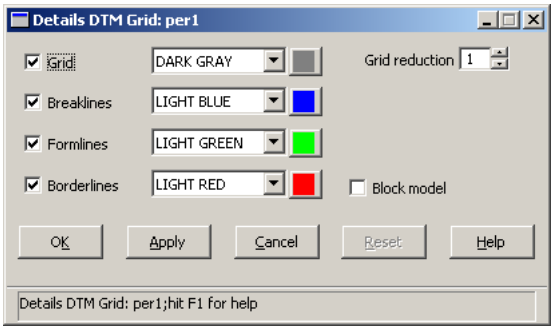


Figure 14.5.: The window *Details DTM Grid ...*

are checked marked, grid- break-, form- and borderlines are strictly considered. They are displayed in the perspective window. In the other case they are not displayed. The visual appearance of each type of line can be modified by selecting colors using the respective selections. Usually each line of the DTM raster is treated by the program. Having a very dense raster or simply for computing time reasons it can be desirable not to work with all raster lines. By indication of the parameter *Grid reduction* the purposeful selection is possible. The checkbox *Block model* turns on or off the block model. The value for the reference height is the minimum elevation value of the DTM.

14.4.3. Silhouettes

The button *Silhouettes...* switches the silhouette lines on (when item *Activate* is selected) or off (when item *Deactivate* is selected). Left-clicking the item (Properties...) opens the dialog window *Details Silhouettes*. The dialogue *Details Silhouettes* set up silhouette lines, which are computed from the grid lines using the actual orientation parameters.

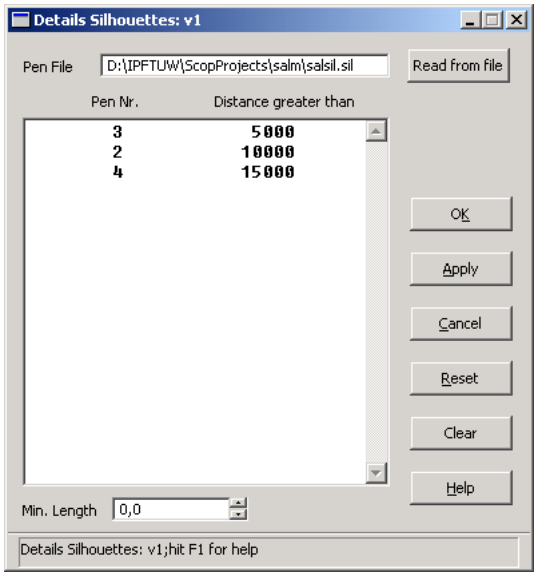


Figure 14.6.: The window *Details Silhouettes ...*

The silhouette lines can be shown in the graphic display in different colours according to their distance from the projection center. The pen numbers are assigned to colours. Silhouette lines, further away than the indicated distance are drawn with associated pen. Another way is to specify a pen-file, that contains lines with pairs of *pen number* and *distance from the point of view*). Up to 100 of such pairs can be specified. The given pairs will be internally be sorted in ascending order by distance beforehand. The name of the pen file can be typed in the text field *Pen file* or it can be browsed for using the file selector (button *Browse...*).

The numeric field *Min. Length* permits to reject silhouette lines, which do not achieve an indicated minimum line length within the perspective view.

14.4.4. Names

The button *Names...* switches the name information on (when item *Activate* is selected) or off (when item *Deactivate* is selected). Left-clicking the item (Properties...) opens the dialog window *Details Names*. The dialog window *Details Names* serves to configure the names information in a perspective view, to identify area objects. The names information is read from a file which have to be specified by text window *Names file*. The name of the file may be browsed using the file selector (button *Browse...*).

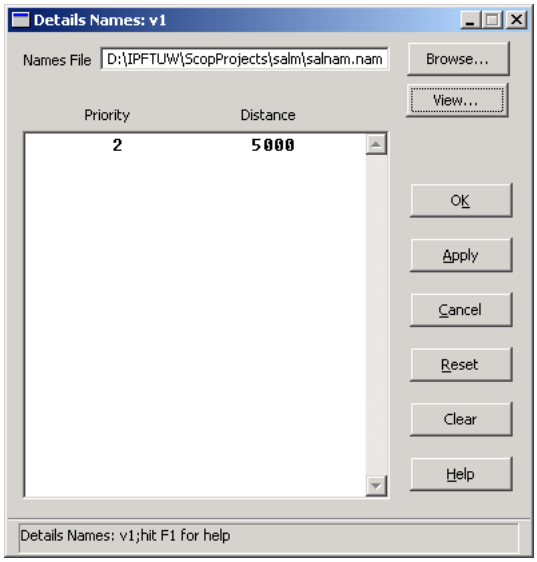


Figure 14.7.: The window *Details Names* ...

The button *View...* opens the Names file. The given file must conform to the following format specification:

Columns	denotation	explanation
1–32	name	text
33–44	east coordinate	floating point value (format F12.2)
45–56	north coordinate	floating point value (format F12.2)
57–68	height	floating point value (format F12.2)
69–72	priority	integer value within range 1 to 99
73–512	perimeter	radius of a circle representing the the named object, or up to 32 pairs of coordinates of polygon points. The given polygon represents the named object. The coordinates are relative to the position of the named object. The values may be free formatted.

A specific distance may be associated with each priority by specifying pairs of priority and distance within the corresponding colums of the dialog window . The distance has to be specified in kilometres.

The criterion Priority decides whether a named object is to be represented or not. A pair of priority and distance is interpreted as follows: if a named object with a given priority is nearer than the specified distance, then it is in principle applicable for the mark. If the named object is visible (at least a small part of the associated perimeter), it is displayed. If the name lies out of distance or is invisible, then it will be disregarded.

14.4.5. Frame

The button *Frame...* switches a heading for the map sheet on (when item *Activate* is selected) or off (when item *Deactivate* is selected). Left-clicking the item *Properties...* opens the dialog window *Details Frame*. In the text window *Title* a heading text to the perspective view may be entered.

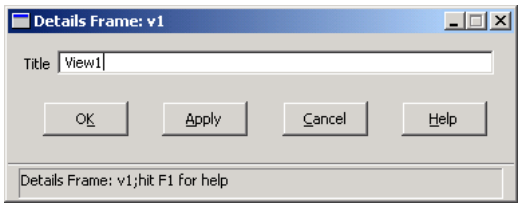


Figure 14.8.: The window *Details Frame*

14.4.6. Additional Parameters

Among them there are numeric fields in *Additional* group from perspective main window: exaggeration, earth curve and refraction coefficient. Supplementary in the group *Image resolution* from the sub-dialog *Details orientation + camera...* the extent of a pixel or the number of pixels for perspective view can be selected.

A commandline / commandfile examples for *Perspective Views* windows are:

Procedures Visualizer:

@AddPerView;	Adding an perspective view
@GridPer;	Display perspective view of DTM grid
@DetOrient;	Define details of orientation
@SilPer;	Display silhoutte lines
@ImagePer;	Display perspective view of raster images

cmL example

```
// Adding an perspective view
@AddPerView;
  Perspectives,
    AddPerView,
      Name=(3DView),
      Comment=(Test),
      OK;
@endProc;
```

cmL example

```
// Display perspective view of DTM grid
@GridPer;
  3DView,
    SelectOverlay=(ens2000),
    Setup=(NW),
    ElevAngle=(-10),
    ViewDist=(1000),
    Exagg 2.0,
    DtmGrid,
      Grid=(1),  ColorGrid=(white), GridRed=(1),
      Break=on,  ColorBreak=(lightblue),
      Border=on, ColorBorder=(lightred),
      Ok;
    DTMGrid=(activ),
    MainGr=(deact),
    Apply;
@endProc;
```

cmL example

```
// Define details of orientation
@DetOrient;
   @GridPer;
   3DView,
     det,
       Auto=(Origin),
       Target 56984.805 64880.545 258.414,
       Rotations 300 -30 100,
       innori 25 50,
       Pixelsize=1,
       OK;
     Apply;
@endProc;
```

cmL example

```
// Display silhoutte lines
@SilPer;
  @GridPer;
  3DView,
    Silh,
      Ok;
    DTMGrid=(deact),
    MainGr=(deact),
    Silh=(activ),
    Apply;
@endProc;
```

cmL example

```
// Display perspective view of raster images
//(Shading and Z Coding)
@ImagePer;
  ens2000/
  Shade display,
  Zcode display;
  3DView,
    SelectOverlay=(ens2000),
    DTMGrid=(deact),
    Silh=(deact),
    MainGr=(activ),
  Apply;
@endProc;
```

Part V.

Theory Manual

15. Introduction

15.1. Preamble

This manual is to provide basic theoretical background to SCOP. The main subject is the digital elevation model with its data structures and methods of its computation (*interpolation*). It also provides information on processes such as isoline (i.e. contour line) derivation.

Traditionally, it is topographic maps to carry information on the terrain surface (hypsometry), and on objects scattered over it (planimetry). Planimetry is made up mainly by traffic routes, buildings and land usage, whereas hypsometry reflects the third dimension of the terrain surface: height or elevation.

With automation playing an ever increasing role it seems to be useful or even inevitable to provide the above information in forms accessible by digital data processing.

In today's data processing technology this information is split into position and elevation. Information on position is integrated into database systems and interactive graphics systems. Hypsometry is subject of other software developed separately: *Digital Terrain (or Elevation) Models*. The reason for this separation is, on the one hand, tradition – but on the other hand also the basically different data structures needed for efficiently dealing with height information. Height information is typified by continuity with local disruptions, best represented as combination of grid data with line structures, which can be related to position information only with considerable difficulties.

SCOP is for computation and application of *Digital Terrain Models* (DTM; the more precise expression *Digital Elevation Model* is also often used with the same meaning).

15.2. Terrain and Model Surface

The term *Terrain Surface* refers to an abstract idealization of the Earth's surface, derived for some specific purpose (Wild, 1983). *Generally speaking* it is the Earth's surface, with the ground's roughness smoothed, and with certain objects on it omitted – the latter to concern vegetation, buildings, and the similar.

The term *Model Surface* refers also to an abstract surface coming as close as possible to the *Terrain Surface* as defined above. It is derived (*interpolated*) of *reference data* as provided by *Data Acquisition*. Reference data describe the *Terrain Surface* with an accuracy and density depending on the method and accuracy of data acquisition, on *completeness* of the latter, on surface *roughness*, on the terrain surface itself (rocks, artificial surfaces), etc.

The process of *Interpolation* usually allows for

- *smoothing* the roughness of the surface, and for
- *filtering* accidental errors in reference data

16. Data Acquisition

16.1. Methods

Data for computing Digital Terrain Models is acquired by methods widely different in their purpose and accuracy, and belonging to various technological fields:

- *Land Surveying*,
- *Aerial Surveying (Photogrammetry, both analog and digital)*,
- *Airborne Laser Scanning*,
- *Remote Sensing* (applying, e.g., satellite imagery)
- *Contour Map Digitization*
- *Echolot recording* (hydrology)
- and numerous others.

One has to differentiate between the *measuring precision of the surveying system* and the *accuracy of terrain modeling*. Especially in terrestric surveys, precision of a few cm can be reached; however, the accuracy of describing the surface is also influenced by point density, vegetation, surface forms, and by user specifications aiming at neglecting small or minute forms (smoothing) – and is thus less accurate.

The accuracy of photogrammetric surveys depends mainly on the flying height (Kraus (1997) and Kraus (2000b)). Stationary measurements of aerial photographs yield a higher accuracy, typically $\pm 0.10\%$ to $\pm 0.15\%$ of it. Dynamic methods, such as scanning of parallel profiles or directly acquiring contour lines in a stereo model, will decrease this accuracy. Especially with increased scanning velocities and for steeper terrain, accuracy can drop to $\pm 0.3\% - \pm 1.0\%$. Low quality photographs will decrease accuracywell below the accuracy of the measuring equipment. Photogrammetric data acquisition applying multiple stereo models can lead to discrepancies in heights between the models, to overlaps, and gaps.

Traditional methods of data acquisition for digital terrain models are often disabled altogether by vegetation, bad visibility and impossibility to reach certain points. *Laser scanning* can considerably easy these problems. In this case it is sufficient if the point to be measured will be reached by a single relatively steep laser ray – whereas in aerial photography one needs two typically less steep rays to rich this same point; in addition, the point has to be sufficiently illuminated by the sun. The accuracy of elevations as measarued by laser scanning is in the range of ± 0.15m to ± 1m (Kraus (2000)).

16.2. Data

16.2.1. Classes of Terrain (Acquisition) Data

Terrain data (also called *Acquisition Data*) can be classified as data *with* and *without additional form information*.

Data Classes *without* Form Information

Data classes without form information are referred to in SCOP as *bulk points*. They can be acquired according to the following two criteria:

- chosen according to the form of the terrain surface, as it is the rule in tacheometric surveys, or
- chosen according to pre-defined point distribution and density. Examples are: grid measurements, data acquired by laser scanning, points along parallel profiles – or just distributed at random. In this case, the form of the terrain surface does not influence which points are chosen, and how many of them.

Data belonging to these classes are often referred to as *Raster-type*: they will be fully replaced by the *DTM Grid* (being in some sense a raster).

Point Classes *with* Form Information

These include:

- Spot heights such as saddle points, peaks and depressions
- contour lines (measured or digitized)
- Break lines describing sharp edges of the terrain surface,
- Form lines (or *Structure lines*). These represent also characteristic linear structures of the terrain – but less sharp than those to be described by break lines. Skeletal lines and mild valley lines are typical examples.

Data belonging to these classes are often referred to as *Vector-type*: they will *not* be replaced by the *DTM Grid* but integrated with it so to form the *Hybrid DTM Structure* of SCOP.

16.2.2. Some Notes on Applying Different Point Classes in SCOP

To understand the meaning of the different data classes in SCOP, refer also to section 18.2.2 describing algorithms, and also to section 18.3. Here, some general remarks follow.

- As already mentioned, vector-type data are integrated with the SCOP DTM structure. Additionally, intersections of break and border lines with the DTM grid becomes also part of the DTM structure. This plays an important role in efficiently handling the form information carried by them when deriving views and other products from the DTM.

- *Break lines* are of special importance since they describe surface edges. This partitioning is employed by partitioning the surface into areas bordered by break lines, which are then interpolated separately (*Interpolation Areas*). Also, there are several means provided to the user to control the influence of break lines on the interpolated surface - such as point densification along them as a pre-processing step, specifying a separate filter-value for this data class, requesting the interpolation of grid-intersections by linear interpolation along the break lines.
- The methods provided to control the influence of *form lines* on the interpolated surface are identical to those for *break lines*.
- *Border lines* can be acquired with or without elevation – but border lines with elevation are by far preferable (for to avoid any extrapolation). There is also a special type of border lines provided to act at the same time as break lines: a case often met in nature.
- In SCOP, *Break lines* and *Border lines* can be coded as open or as closed. In case of closed (rather: *to-be-closed*) lines the program will connect the first point of the line with the last one, and this connection is then treated as part of the line. If the first point of a line is identical with its last point – i.e. if that point is acquired twice, – then the line can be coded as open, and it still will be treated properly.
- *Spot heights*, being vector-type data, are also integrated into the DTM structure. While interpolating the terrain surface they can be given – as a class – a special filter value (definition of filter values see 18.2.2). Their elevation can be labeled on the contour map.
- *Off-terrain points* can be input into SCOP also. This type of data will not influence interpolation; it is just for graphical representation (e.g. church crosses).
- *Points at Random* are scattered at random over the surface. There is no requirement whatsoever concerning the sequence of their acquisition.
- *Regular Profiles* are treated the same as points at random – except for in special cases, when the user applies *Affine Filtering* to them: the algorithm can derive the corresponding parameters of data coded as *Regular Profiles*.
- *Contour Lines – Digitized or Measured* are considered to be *raster type* data, and correspondingly will be replaced by the DTM grid as interpolated. The information on surface shape and orientation carried by them can be extracted from them by special processes (see 16.3).

16.3. Structure Extraction; Pre- and Post-processing

There are special methods integrated into SCOP++ for such purposes:

- Applying specific algorithms of triangulation (D. Heitzinger and H. Kager (1998) and D. Heitzinger and H. Kager (1999)), bridging over gaps in reference data can be performed in a pre-processing step to DTM interpolation. This process can be considered as homogenization of data distribution – achieved by filling larger gaps in the cloud of acquisition points.

- Applying another form of the triangulation mentioned above, form lines (i.e. structure- or skeleton-lines) can be derived of digitized or measured contour lines, and included into the data set so to aid DTM interpolation.
- The DTM as interpolated, e.g., of pre-classified laser scanning data, can be subjected to post-processing by rain-simulation (considered to be a way of *surface exploration*) resulting in vector-type information on probable situation of water flow lines, break lines and form lines. With these integrated with the original set of data, a surface improved in a geomorphological sense can be attained by repeating the interpolation.

17. The SCOP Digital Terrain Model

17.1. General Introduction

Digital Terrain (or: *Elevation*) Models are similar to traditional contour maps in their common task to represent the third dimension of the terrain surface. The graphical solution applied in mapping has to be transformed into a *numerical solution suited for digital data processing*.

A digital terrain model is typified by its structure and its method of computation.

The prerequisite for effective work with digital elevation models is an efficient structure. The user should not be required to have much knowledge of this structure, although a basic knowledge of it will be certainly useful when analyzing results.

Various solutions are there to describe the terrain surface by digital models. The following principles can be distinguished:

a) Partitioning into Interpolation Areas so that the description of the surface is partitioned in individual interpolation areas. The partitioning can be done by
   - subdivision of the area into regular interpolation areas of suitable size (usually rectangles)
   - a net of triangles with vertices that are reference points so that the interpolation areas consist of triangles of varying size and shape.

b) Description of the Surface of the Interpolation Areas
   - by some mathematical function
   - by a regular grid of heights derived from the mathematical function.

   The mathematical function to describe the surface can be determined by interpolation, approximation or prediction. One can distinguish between
   - simple and fast interpolation methods
   - gliding polynomial surfaces (ineffective if a great number of heights has to be calculated, and unsuitable for high precision)
   - precise interpolation methods such as by splines, by linear prediction, or by finite elements.

17.2. Basic SCOP-specific Concepts and Terms

SCOP is based on this *GENERAL CONCEPT:*

- Create a Digital Terrain Model to suite as well as possible the products to be derived of it. For this purpose, apply filtering of accidental errors in reference data, surface

smoothing (we cannot state: generalization), eventually surface modeling and the similar;

- derive of this model the products as requested by efficient and simple methods. There must not be any need for filtering, smoothing etc. – all this has to be done in creating (*interpolating*) the model.

An example of the above is contour line (*isoline*) derivation: contour lines as derived of the SCOP DTM, provided that the resolution of it has been chosen properly (see 17.8), will not be smoothed, lines will not intersect, and adjacent lines will present a nice harmony corresponding to the smooth DTM surface.

The Hybrid DTM Structure

SCOP uses for the description of the terrain surface a subdivision into rectangular interpolation areas of constant size (*Computing Unit: CU*). The surface in each rectangular piece is described by a mathematical function. Discontinuities in the surface (represented by break lines) necessitate the use of more than one function. The mathematical function or functions are then used to derive a rectangular grid of heights, the *DTM grid*, providing the discrete description of the surface; the DTM grid replaces the interpolation function, and is to be considered as a discrete form of it. In addition vector-type reference data are integrated with the grid, in order to be able to represent them exactly (rather than just with the accuracy of the DTM grid's resolution); this yields the *hybrid DTM structure* – a specialty of SCOP.

The elevation of any randomly placed point can be interpolated based on the DTM grid. Furthermore it is possible to derive lines consisting of points of equal height (referred to in SCOP as *isolines*). The DTM grid also allows the computation of slope and curvature of the terrain surface.

In `SCOP V3.5`, the computing units (*CUs*) are also the storage units: after DTM interpolation, they carry the corresponding portion of the grid. Applying however SCOP.PAK (the Programmer's Application Kit), computing units will be of variable size depending on reference data distribution; on the contrary, *storage units* in this case are of the same standard size all over the DTM; they are derived of the CUs, and are *not* identical with them.

Some Important Terms and Concepts:

- DTM limits
- Computing unit (CU)
- Overlapping area
- Averaging at the edges of computing units
- Empty computing unit
- Grid step
- Line/Grid intersection
- Line concatenation

In the following sections, these terms and concepts will be explained in some detail.

17.3. DTM Limits

The DTM is computed within rectangular limits with sides parallel to the co-ordinate axes. If nothing else specified, the maximum extent of reference data will be taken as default to limit DTM computation. Other values can be entered numerically or digitized by the user in the SCOP++ window *Limits*, or specifying corresponding parameters in `SCOP V3.5`.

17.4. Partitioning into Computing Units

The DTM area is subdivided into equally sized rectangular Computing Units (CU). Under `SCOP V3.5`, the computing unit is an interpolation unit as well as a (*random* or*direct* access) storage unit. Under the SCOP Programmer's Application Kit, CUs for interpolation and for storage are different.

Computing units can be regarded as elements of a matrix. Each CU is marked by its index Northing/Easting. The lower left corner of the CU 1/1 corresponds to the lower left corner of the DTM limit. Because the edges of CUs along the upper and right side of the DTM generally do not coincide with these edges of the DTM limits, such CU's will be computed and stored beyond them.

SCOP terminology distinguishes between *Net Computing Units (CUs)*, and *Gross Computing Units (GCUs)*. In interpolating the surface of a *Net Computing Unit*, reference data within the *Gross Computing Unit* will be applied (see 17.5). The maximum area that can be covered by a GCU equals 5*5 CUs – with the CU to be interpolated in the center of this 5x5 area.

The size of the computing unit must be an integer multiple of the grid step of the DTM. The number of grid lines within a computing unit has to be 3 through 49, where the number of grid lines includes the limiting lines also (i.e. a CU two grid steps wide contains 3 grid lines).

The size of computing units can either be specified explicitly by the user as the number of grid lines in it, or be automatically determined by the program. Use explicit specification if

- constant DTM structure is important for several different models
- experimenting with specific cases of DTM interpolation, e.g. very flat terrain, extreme variations of reference data distribution, intense smoothing.

The automatic determination of CU size is an iterative process. It is aiming at a size to contain a specified average number of reference points in the computing units. The algorithm checks on every iteration the actual number of reference points per CU concerning average and maximum. This iterative process is not necessarily convergent in cases of extremely irregular data distribution: the change in the size of CUs between iteration steps leads to shifting the position of CUs, and thus to unexpected average and maximum number of points in the next iteration. To avoid any endless loops, a maximum number of iteration steps can be specified (the default being 2).

Experience with reference data stemming from photogrammetric or terrestrial acquisition shows an average number of points of 9 to 16 per CU to be advantageous, since this

corresponds to the optimal computation time. The optimal relation 1:4 through 1:16 between the number of reference points and the number of grid points results in 36 through 256 grid points per CU. – In case of data acquisition by laser scanning or image matching, however, the number of reference points is usually considerably higher than the number of grid points. This circumstance spells the necessity of special structural considerations for these methods.

Extremes *irregularities in data distribution* cannot be handled just by choosing some compromise in CU size. In SCOP++ there are special methods available to deal with this problem (see 16.3). This has much to do with generating empty CUs (see 17.7).

Furthermore, *surface continuity along the edges of adjacent CU's* has also much to do with CU size (see section 17.6) – especially if *considerable smoothing* is required.

17.5. Overlapping Areas

Once CU size and the lower left corner of the DTM are defined, the reference data can be sorted into the *Gross Computing Units (GCUs)* (17.4).

Points contained within the *Net Computing Units (CUs)* do not represent sufficient information for the description of the terrain surface within them. One also has to consider points around the CU. These are considered to belong to the CU's *overlapping area*.

Points close to the sides of the CU (1st order neighbors) are absolutely important to avoid extrapolation at the edges of the CU. The next layer of points (2nd order neighbors) provide information about the curvature of the surface within the CU. Very far points however do not influence the surface within the CU.

The user can choose between small, medium, large, xlarge, and xxlarge overlapping. The small overlap corresponds to searching for 1st order neighbors only, large overlap also searches for 2nd order neighbors. Medium overlap is a compromise between small and large overlap. xlarge and xxlarge overlaps are useful for special purposes (see, e.g., 18.2.2

The program is proceeding as follows. The vicinity of the CU is subdivided into 12 sectors out of which a number of points are successively extracted. The search continues for a maximum distance of 2 CU's. The number of points to be extracted results from the chosen overlap and the average point density in the gross computing unit (GCU).

The measure of overlapping as chosen has a decisive effect on the continuity of the DTM surface at CU edges. This continuity also depends on point density and distribution. When modeling flat terrain, it is advisable to choose xlarge or even xxlarge overlapping. Regular and dense point distribution will yield fine results at medium overlapping also. For a solid regular quality one rather should apply *large* overlapping. For intense smoothing one should choose larger overlapping, otherwise surface continuity will suffer considerably.

17.6. Averaging at CU Edges

Common edges of adjacent CUs will be computed twice. The final values along these lines are gained as simple average.

Any discontinuities between the surface patches as computed for adjacent CUs are thus averaged over a distance of two grid steps. The smoothing effect is smaller for small grid steps than for large ones. Unfavorable reference data distribution, rough errors in data acquisition concerning both accuracy and point location, and also problematic terrain forms may result in CU-edges becoming visible, e.g., in hill shading and other views derived of the DTM. This is especially obvious for small grid steps combined with large CUs.

On the protocol file summarizing results of the DTM interpolation, statistics concerning discrepancies along CU edges is written. This statistic is somewhat distorted by it including discrepancies along the edges of CUs at the DTM's limits, with parts outside of such limits and therefore extrapolated. Nevertheless, for large models with numerous CUs, this statistic characterizes well the accuracy of the model.

17.7. Empty Computing Units

Although the matrix of CUs logically contains all of them within the rectangular area defined by the DTM limits, those CUs without sufficient reference data for their interpolation will be left *empty*, and handled as such by any process accessing the DTM.

If the *Gross Computing Unit (GCU)* does not contain at least 3 points *enclosing* the CU, the CU is not computed and thus becomes an *empty* one.

This ensures that CUs not supported by reference data are never calculated. The algorithm results in empty clusters containing always more than 3*3 CUs.

17.8. Grid step

It is important to understand the meaning of the grid step to be the resolution of the surface representation: it is assumed that within a grid element with the size (grid step x grid step), elevations can be computed by simple linear interpolation, without damage to accuracy. There is, however, a further point to this: the grid step is representing DTM resolution in the XY plane also. This circumstance can pose problems in different products derived of the DTM, when the grid step is chosen to be too large for it (see for instance 22.

If break lines of the terrain surface are acquired, then – since break lines are integrated into the Dtm grid – they do not have to be specifically considered when choosing an appropriate grid step. In the opposite case – including also *laser scanning*, – a smaller grid step has to be chosen, otherwise these lines cannot be reflected properly by the model surface.

Choosing the Grid Step The user should specify the resolution for the DTM to represent the surface of the terrain, so to suite certain purposes: in addition to reference data accuracy and completeness, the accuracy and completeness of terrain representation by the SCOP DTM is *basically* depending on its grid density.

There are two major points to consider when selecting the grid step for creating digital terrain models to serve some general or special purpose:

- *The accuracy and completeness of information carried by the reference data.*. It is depending on
  - the amount of detail that was recorded or ignored in data acquisition, and
  - on the accuracy of the data acquisition technique applied.

  With bulk reference data replaced by the Dtm grid, this information can be partially lost when grid resolution cannot reflect it properly. Loosely speaking, for accurate data this will require a grid denser than the average density of reference data. For reference data with large accidental errors, however, the contrary might be true.

  In any case, efficiency and economy will limit requesting high grid density – notwithstanding the fact that it is reference data density having the decisive impact on interpolation time (see section 18.2.2). Altogether one is seeking here a suitable compromise.
- *Fit best the product to be derived of the DTM* - e.g.
  - contour map of a certain scale with hill shading and Z-coding,
  - volume computation for engineering purposes.

  The most important – and *limiting* – question here is to decide, whether the accuracy and completeness of reference data can support the given purpose.

  Only if the answer to the above question is yes can we go on to consider other points in choosing a proper grid step:
  - *smoothing* of terrain roughness – or concerning grid step rather: *neglecting* it – may be an important consideration. A good example is applying *laser scanning*: in open areas reference data are both of high quality and of extreme density. Data will reflect minute structures of the surface: a completeness of information by far exceeding requirements in any usual case. Reason to choose a grid step often considerably exceeding the average point distance in reference points.
  - *graphical quality of certain products* may have much to do with choosing a grid step. E.g., for fine graphical quality of contour lines derived of the Dtm, the grid step of the latter should not exceed some 1.5 mm in the sacel of the map, at least in topographic mapping (this value may be doubled in very large scale maps serving engineering purposes). A smaller grid step may be needed for hill shading.
  - *Numeric products for engineering purposes* are often derived of dense and precise reference data sets covering areas limited in their size. The best examples belong to *DTM Algebra*, e.g. volume computation. In this case, information carried by the reference data set will decide the grid step to be applied: the latter is going to be $1/2$ or even $1/4$ of the average reference point distance.
  - *averaging along CU edges* may also influence the grid step to be chosen (17.6)

  The above considerations are typical for most practical applications, and correspond to considering the reference data rather than the DTM as the main carrier of terrain surface information – and seeing the DTM as an intermediate product of computation, structured and derived to correspond to the product aimed at.

17.9. Line/Grid Intersections

In addition to the grid points, intersections of break lines, form lines and border lines with grid lines are stored in the DTM. Thus it becomes possible to rigorously consider break lines as terrain edges and border lines. This *hybrid Dtm structure* contributes also substantially to the consistency of form line representation.

Computation of line/grid intersections is described in chapters 18, 19 and 21. In addition, for linear prediction, the special case of linear interpolation along break lines and form lines is implemented (see 18.2.2)

Line/grid intersections are vitally important for the programs using the DTM. They are considered in deriving isolines, cross-sections, digital slope models, perspective views, computing DTM Algebra, etc.

18. DTM Interpolation by Linear Prediction

18.1. The Task

The task consists of development and realization of an algorithm to describe terrain surfaces with high accuracy but with computation times being within acceptable limits. Here one has to consider the characteristics of the the surface as well as those of surveying method used.

The aim for a very precise description of the surface is that contour lines shall comply with the topographic demands. During the last years a significant increase in performance was achieved, but the high topographic demands are not always fulfilled, if difficult terrain structures are not surveyed optimally.

Terrain surfaces are according to section 15.2 an idealization and an abstraction of the Earth's surface. They tend to be smooth with continuous differentials over large areas. There may be area type structures as well as linear structures. Further one has to differentiate between large forms and small forms. Large forms represent the terrain surface on a large scale, while small forms are local departures of the general surface.

Discontinuities are existing in the form of surface edges and escarpments, where surface edges are discontinuities of the first differential and escarpments are discontinuities of the functional values themselves. These discontinuities do not directly exist on the Earth's surface, but they are generated by scaling down of the surface and the projection of the surface onto the horizontal plane. Rivers in mountainous regions lead to surface edges. Artificial structures like retaining walls and dams can also lead to surface edges and escarpments.

The theoretical basis of linear prediction is presented in various publications by Kraus, Assmus and Wild (see Section 23 Literature). Linear prediction corresponds to the statistical estimation method *Krige* often applied in geo-sciences (Kraus (1998)).

The following sections of chapter 18 contain a comprehensive overview of the solution as realized by SCOP.

18.2. The Solution

For the precise description of the terrain surface the program system SCOP uses linear prediction. The surface in each computation unit, or if break lines are encountered, each interpolation area, is described by a prediction function. The heights of the grid points and grid section points in the DTM are calculated from this function. The algorithms

used to compute the prediction function in dependence of the prediction parameters will be described in this section.

The computation of the DTM is done separately for each computation unit. According to the occupation matrix the strips are evaluated from left to right, starting at the bottom strip. The evaluation of each CU can be subdivided into the following steps, the most important steps will be explained in the next section:

a) Preparation of the DTM Computation:
   - Reading of points lying in the CU and its overlapping area
   - Variation of grid step, if the criterion point density was selected
   - Insertion of break lines and border lines into the DTM grid
   - Generation of interpolation areas if break lines are present
   - Densification of break lines and form lines
   - Sorting of points into the interpolation areas
   - Relating the grid points and grid sections to their interpolation area
   - Evaluation of trend surfaces

b) Evaluation of the Prediction Function and Computation of the Heights of the Grid Points and Grid Sections
   - Evaluation of the prediction parameters
   - Computation of the prediction function
   - Checking of filter values
   - Computation of the heights of the grid points
   - Computation of the heights of the grid sections
   - Computation of the heights of spot heights
   - Variation of grid step if slope or curvature were selected as criterion
   - Storing of CU

18.2.1. Definitions and Descriptors

The following Descriptors are of fundamental importance for the solution of the problems. They will be explained in the following sections.

- Interpolation areas
- Break line combination
- Pseudo lines
- Break line densification
- Trend surface
- Covariance function
- Prediction function
- Filter values (mean and maximum)
- Elimination of scanning errors

18.2.2. Algorithms

Generation of Interpolation Areas

If a CU does not contain break lines (surface edges) then the terrain surface can be represented by one continuous function. If it does contain break lines the surface has discontinuities so that it is advantageous to use a straight line as covariance function (see section 18.2.2) as well as to subdivide the CU into interpolation areas according to the break lines. Each interpolation area represents an independent prediction range, and neighboring interpolation areas are only connected through the common points along their break line.

For the generation of interpolation areas it is sensible to differentiate and classify break lines depending on their situation as follows:

- Break lines starting and ending outside the CU: Type A
- Closed break lines: Type B
- Break lines that start and/or end within the CU: Type C

Break Lines of Type A As long as break lines start and end outside a CU, a subdivision into interpolation areas can be easily done, because here it is only necessary to check whether a point lies to the left or to the right of the break line.

Break Lines of Type B Closed break lines also lead to a clear partition between the areas lying inside and outside the line.

If an embankment's upper and lower edges have been defined as a closed break line care has to be taken so that the areas above and below the embankment are not connected. Otherwise a wave-like surface is generated close to the embankment edges, because the points of both edges are interpreted as belonging to the same interpolation area.

A closed break line can thus be easily recognized. If a closed or nearly closed break line was coded as open ended then it will be closed, if the distance between two points at or near the end of the line is less then a definable value, or if lines are intersecting.

Break Lines of Type C The program tries to connect free ending break lines with other break lines. Firstly the possibility is checked to connect free ending break lines among each other, if this is not possible the other break lines are checked.

A connection is made if:

- the distance between two points on two lines is less than the *search-radius for line networking*.
- the perpendicular distance between a point and a second line is less then a given radius.
- two lines are intersecting.

Here the program starts with the last point of the free ending line and continues towards the middle of the line for the following points on the line to check if a connection is possible.

The Generation of Pseudo Lines

Once the possible connections are found pseudo lines are generated. Pseudo lines are of no importance to the user of the program. They are used solely as an internal aid to enable the generation of interpolation areas, according to the running of the break lines, for which the surface is represented by an individual function.

Pseudo lines start or end in the corners of the gross CU. In addition they have to run from left to right completely through the CU, so that a decision can be made whether a point lies above or below the pseudo line.

Lines of type A can thus be changed into pseudo lines simply by adding 2 gross-CU-corners.

Free ending lines, for which according to 18.2.2 connections were found, are put together and use parts of other lines such, that pseudo lines are also generated.

Closed lines are logically dissolved and are also changed into pseudo lines.

Pieces of lines without connection or those lines lying completely outside CU are ignored during the generation of interpolation areas. These lines are sorted into the interpolation areas although and are thus used during the evaluation of the prediction function.

The Insertion of Points into the Interpolation Areas

With the aid of pseudo lines it is possible to decide on which side of the pseudo line a given point will fall, and thus by consideration of all pseudo lines it becomes possible to give the point an identification number corresponding to the number of its interpolation area. This number changes when stepping over a pseudo line.

While pseudo lines form a rigorous boundary for bulk points, it must be guaranteed that points on a break line will fall into both interpolation areas to the left as well as to the right of the line. The sorting of points contains round-off problems. A point lying exactly on the break line can either fall into both interpolation areas, or only into one of the two areas.

In the case of break lines there is the condition that, since the two adjacent prediction functions should join up, break line points must fall into both areas to the left and to the right of the break line. Because of this break line points are displaced by a small amount along the angular bisector formed by the intersection of the two break line intervals.

The identification of the correct interpolation area must also be performed for the grid points and for the grid sections (intersections of break lines with the grid lines). Grid section must also be present in both interpolation areas, their predicted heights are calculated from both adjacent prediction functions and the average is taken. The same applies to grid points lying in two interpolation areas.

Trend Surface

While originally the *trend surface* was used to split the systematic part from the stochastic part in a given set of data, in order to analyze the reduced set of data with statistical methods, the trend surface now serves a slightly different purpose. It is used firstly to

center the set of data in order to calculate their variance. The variance allows certain conclusions about the size of the data. Secondly approximately plane terrain surfaces can be approximated quite well by this method.

The use of a curved trend surface bears the danger to superimpose the curvature of the trend surface onto the predicted surface.

If one for instance approximates a cut-in road by a curved trend surface then the curved trend surface will follow the depression. The distance between the horizontal road surface and the trend surface is thus no longer linear. If there are no surveyed points in the center of the road the prediction between the road edges will be linearly interpolated. By adding the linear part to the curved trend surface a curved road surface is generated.

To avoid this effect of superimposing trend forms onto the prediction function a plane trend surface is suitable, either in the form of a *horizontal plane* or a *tilted plane*.

A connection between neighboring trend surfaces is not necessary since the predicted surface should be defined by the surveyed points and not by the chosen trend.

If one considers especially road surfaces, embankments, retaining walls and dams the surfaces can often be approximated optimally by choosing a tilted plane, if the surface edges are defined by break lines and each interpolation area is evaluated by a tilted plane. On the other hand predictions have difficulties in displaying tilted plane and the result is the formation of small waves and terraces between the reference points.

It can be concluded that the use of tilted planes in each interpolation area for the description of approximately plain parts of the terrain surface offers some advantages because the terrain surface is well approximated by this. In undulating areas there is hardly any difference between the use of a horizontal plane and a tilted plane as the trend surface.

The tilted plane is secured to avoid excessive sloping. The check is performed through the points used to derive the tilted plane and those values taken on by the tilted plane within the CU. If the slope of the tilted plane is considered too large the trend surface is replaced by a horizontal plane in that interpolation area.

Break Line Densification

Point densification along break lines is achieved by linear interpolation of additional break line points between the measured points (*Line Densification: Interval for break line point densification*). This has to be done if the break line points are far apart.

Bbulk points lying close to a break line will disturb the path of the interpolated break line if they have a differing height. Thus the density of the break line points should correspond to the density of bulk points in any case. If the break line is used to cut out a narrow, curved interpolation area an even higher density is recommendable.

The densification is activated if the distance between two break line points is larger then the given densification interval. The densification distinguishes between break lines inside and outside the net-CU.

Inside the CU the intersections between the break lines and the grid lines are used as reference points if the distance between two break line points is larger than the densification interval.

Outside the CU an increasingly larger interval is used depending on the distance away from the CU border, starting with the given densification interval.

Form Line Densification

The call to form line densification results in the insertion of additional linearly interpolated form line points(*Line Densification: Interval for Form line point densification*). The densification is performed in the same manner as for the break lines. It results in a better fit of the prediction function on the form lines and is thus recommendable under most circumstances, unless the area was already surveyed with a high point density.

Linear Interpolation of Break Lines and Form Lines

There is a possibility to insert break lines and form lines unchanged into the DTM by calling a linear interpolation of break lines and form lines(Special cases: Linear interpolation of line/grid intersections). Here the grid section of form lines and grid lines are interpolated linearly between the measured heights of the break lines and form lines.

It is advisable in this case to choose a high densification of break lines and form lines as well to smooth out differences to closely neighboring bulk points and to enable the use of the interpolated points as reference points.

The Prediction using Linear Combinations of Covariance Functions

The prediction used in program system SCOP is a linear prediction through linear combinations of covariance functions. It includes the filtering of random errors and in this same process the smoothing of surface roughness; if specified by the user, *Affine Filtering* will be applied. The following prediction parameters are of central importance:

- Covariance function
- Covariance function parameter
- Affine transformed covariance function
- Aimed mean filter values
- Maximum allowed filter values
- Surface smoothing
- Elimination of scanning errors

Descriptors The points measured during the survey are called reference points for the prediction.

Given are n reference points with plane co-ordinates u, v and the corresponding height s. Index j represents a general reference point j with corresponding co-ordinates $u[j], v[j]$ and $s[j]$.

The prediction function serves to describe the terrain surface. It describes the z-values in dependence of the surface parameters x and y. The co-ordinate systems u, v, s and x, y, z are identical. They are kept apart to distinguish between the reference co-ordinates and the co-ordinates of the prediction function.

The Covariance Function Each reference point j has a corresponding covariance function $g[j]$:

$$g[j] = f(x, y, u[j], v[j], s[j])$$

x and y are the surface parameters and $u[j], v[j], s[j]$ are the co-ordinates of reference point j. Possible covariance functions are:

Bell curve: $g[j] = 1/(1 + d * d/(m * m))$

with $d = SQRT((x - u[j]) * (x - u[j]) + (y - v[j]) * (y - v[j]))$
and $m =$ covariance function parameter

Straight line: $g[j] = 1 - ABS(d/m)$

with $d = SQRT((x - u[j]) * (x - u[j]) + (y - v[j]) * (y - v[j]))$
and $m =$ covariance function parameter

Geometrically the covariance functions are representing rotation surfaces, with the axis of symmetry at the corresponding reference point. The function thus only depends on the distance d from the corresponding reference point.

The covariance function parameter m is a scaling factor between the covariance function and the plane co-ordinates x, y. An increase of parameter m results decrease of slope of the covariance function.

The Linear Combination of Covariance Functions If each reference point has a corresponding covariance function centered at its reference point, the resulting prediction function is a linear combination of the covariance functions

$$z = a[1] * g[1] + a[2] * g[2] + \cdots + a[j] * g[j] + \cdots a[n] * g[n]$$

or written in matrix notation

$$z = \boldsymbol{f} * \boldsymbol{a}$$

with $\boldsymbol{f} = [g[1]g[2] \ldots g[j] \ldots g[n]]$
$\boldsymbol{a} = [a[1]a[2] \ldots a[j] \ldots a[n]]^T$

The prediction function describes the terrain surface within a CU or within an interpolation area.

The covariance function g contains the surface parameters x and y. The coefficients a used for the linear combination are the unknown cluster parameters of the prediction function. A general combination of reference points has exactly one set of corresponding cluster parameters. The unknowns a can also be interpreted as scaling factors with which the covariance functions (rotation surfaces) have to be multiplied to generate the prediction function.

The Determination of the Cluster Parameters *a* The determination of the cluster parameters results from the condition that they have to fulfill the reference points. The resulting condition equation for each reference point j is

$$s[j] = a[1] * g[1](u[j], v[j]) + a[2] * g[2](u[j], v[j]) + \cdots + a[n] * g[n](u[j], v[j])$$

where $g[1](u[j], v[j])$ is the value of the covariance function $g[1]$ at the point with coordinates $u[j], v[j]$, and is thus the value of the covariance function at point j.

This leads to a symmetrical system of linear equations. The number of conditions is equal to the number of unknowns as well as the number of reference points. The symmetry results from the fact that

$$g[j](u[i], v[i]) = g[i](u[j], v[j])$$

which means that covariance function $g[j]$ of the corresponding point j takes on the same value at point i as covariance function $g[i]$, which is the corresponding function of point i, will take on at point j.

The system of equations in matrix form is

$$\boldsymbol{F} * \boldsymbol{a} = \boldsymbol{s}$$

$$\text{with} \quad \boldsymbol{F} = \begin{pmatrix} g[1](d[1,1]) & g[2](d[1,2]) & \dots & g[n](d[1,n]) \\ g[1](d[2,1]) & g[2](d[2,2]) & \dots & g[n](d[2,n]) \\ \vdots & \vdots & \ddots & \vdots \\ g[1](d[n,1]) & g[2](d[n,2]) & \dots & g[n](d[n,n]) \end{pmatrix}$$

$d[j,k] =$ distance between the points j and k
$\boldsymbol{a} = [a[1]a[2] \dots a[n]]^T$
$\boldsymbol{s} = [s[1]s[2] \dots s[n]]^T$

The value of the general element j,k in $\boldsymbol{F}$ results from the distance between points j and k, which in turn is the parameter for the corresponding covariance function.

The amount of work needed for solving such a system of equations increases with the number of unknowns as

$$\text{amount} = n^3/6 + n^2/2 - n \cdot 2/3$$

This is the reason why the evaluation of a prediction function from a larger amount of reference points should be avoided. From experience the best computation times result from about 9 - 16 points per net CU.

The amount of work needed for the computation of a DTM is:

amount = (number of CU's) * (average amount to solve a system of equations per CU)

The solution of a symmetric system of equations takes only about half the amount in comparison to an asymmetric system. The advantages of asymmetric systems are so small, that the double amount of computations needed is no point of discussion.

Determination of Predicted Values After solving for the unknowns a it is possible to calculate the functional value for any point with plane co-ordinates $x[i], y[i]$ by solving

$$z[i] = a[1] * g[1](x[i], y[i]) + a[2] * g[2](x[i], y[i]) + \cdots + a[n] * g[n](x[i], y[i])$$

in matrix notation

$$z = \boldsymbol{a} * \boldsymbol{f};$$

$$\text{with} \quad \begin{aligned} \boldsymbol{a} &= [a[1]a[2] \ldots a[n]] \\ \boldsymbol{f} &= [g[1]g[2] \ldots g[n]]^T \end{aligned}$$

Thus one has to compute the distances from the point to be interpolated to the reference points, with these the g values can be taken from the covariance functions, by multiplying these with coefficients a and adding up the final height is obtained.

The amount of computations to be done depends on the ratio between the number of grid points and the number of reference points and it takes about 10 % to 20 % of the total computations needed for solving the systems of equations.

The Prediction with Filtering The evaluation of the prediction function described above only works under the condition that there are no two points with common plane co-ordinates x,y. If there are two points with identical x,y co-ordinates, the system of equations becomes singular, i.e. there is no unique solution, because there are two identical rows in the matrix. This problem also arises, owing to round-off errors, if points are lying closely together.

Although the problem does not only exist under the above mentioned extreme conditions. If closely neighboring points have differing heights the prediction function may swing out, which is not desirable for the description of terrain surfaces.

The swinging-out effect can be avoided, by introducing the possibility of surface smoothing and filtering. The determination of the unknown cluster parameters is changed from solving a system of equations to an adjustment problem by observation equations. The residuals of the observation equations are taken as the differences between the reference values and the predicted values. The unknown cluster parameters are introduced as observations with expected value 0. Since slope and curvature are depending on the cluster parameters a, the minimization of the cluster parameters results in a surface smoothing. Depending on the weight of the observation equations the prediction function will closely approximate the reference points or bigger residuals may result in favor of a smoother surface.

In comparing the generated system of normal equations to the original system of equations it becomes apparent that they differ mainly in the elements along the main diagonals of the equation systems. It can be shown that a smoothing can also be achieved by manipulation of the main diagonal elements of the system. If a chosen value is added to a main diagonal element the prediction function will depart from the corresponding point in favor of a smoother surface. Both the swinging-out effect of the prediction function, and a qualified smoothing of the surface – corresponding also to filtering – can be controlled by manipulation of the main diagonal elements. See Kraus (2000) describing

the specifics of manipulating elements on the main diagonal for filtering accidental errors in reference data. – Specifying parameters for this manipulation is done in SCOP++ under *Stochastic Filtering*, parameters in column *Aimed at: bulk points, spot heights, form lines, break lines*.

The Influence of Covariance Function Parameters The covariance function parameter is a scale factor between the co-ordinate system of the reference points and the covariance function. The covariance function parameter determines the rate of decrease of the covariance function with increasing distance, i.e. it determines the value of the covariance function in the neighboring points. A large covariance function parameter leads to a slowly decreasing covariance function and thus to larger values of the off- diagonal terms. The larger the covariance function parameter is chosen the higher the dangers of swinging-out and singular systems of equations become.

While the covariance function parameter for the straight line as covariance function has hardly any influence on the prediction function, the situation changes for the bell curve as covariance function. A covariance function parameter corresponding approximately to the average point distance is called an average covariance function parameter. Small parameters are in the range smaller than 1/3 and large parameters are at least by a factor of 5 larger than the average.

The following influences can be observed:

- A very small covariance function parameter leads to a decrease of the prediction function between the reference points.
- For a regular distribution of reference points an average or larger covariance function parameter leads to regular convex prediction functions.
- For irregular reference point distributions an average covariance function parameter may already lead to a swinging-out of the prediction function.
- Filtering and smoothing become stronger with increasing covariance function parameters with a constant change of the elements of the main diagonal of the equation system.

The Use of Affine Transformed Covariance Functions Affine Filtering corresponds to performing linear prediction using elliptically rotated bell curves as covariance function. The ellipse can be produced by an affine transformation of the rotational circle – or, and this is what actually happens, by applying the corresponding affine transformation to the reference points before the filtering (thus moving them closer to each other in one direction) before the interpolation, and moving them back by performing the inverse transformation after it.

The parameters of such affine transformation will be derived based on the three values entered to the numeric fields *Direction, Profile Distance, and Point Distance* – another image of this process. The first value determines a direction (as the angle in grades between the East and the direction, counter-clockwise), the second parameter influences smoothing *across* this direction, and the third parameter determines smoothing along it (think of both axes of an ellipse):

- *Increasing* the value for *profile distance* results in a smoother surface across to the *Direction* (of the "profile");
- increasing the value for the *Point distance* (along the "profile") results in a smoother surface in profile direction.

Affine Filtering is performed for each CU individually. The covariance function in such computing units must be specified as Bell Curve (i.e. if *Adapting* has been chosen, Affine Filtering will not take place in CUs containing break lines).

If Affine Filtering in some computing unit is not successful, standard linear prediction will be performed.

*In a special case, when data patterns correspond to a grid or parallel profiles, and are coded as such, then the program can derive values for the parameters as described (*Direction, Profile Distance, and Point Distance *automatically:*

- If numeric field *Direction* is given a negative value, the program will attempt to derive profile direction from the reference points.
- If the values for numeric fields *Profile Distance and/or Point Distance* are specified as 0, the respective values will be derived from the reference points.

For this to function in a computing unit, it has to contain at least two profile sections with three points in each.

Affine transformed covariance functions are mainly used for the elimination of scanning errors. These are also advantageous if the density of reference points is different along two perpendicular axes.

Affine transformed covariance functions are defined by a direction r, a covariance function parameter ma in the direction of r, and a parameter mb perpendicular to the direction r.

$$g[j] = 1/(1 + (sa * sa)/(ma * ma) + (sb * sb)/(mb * mb))$$

$$\text{with} \quad sa = (x - x[j]) * \cos(r) + (y - y[j]) * \sin(r)$$
$$sb = (x - x[j]) * \sin(r) - (y - y[j]) * \cos(r)$$

The intersection of the affine transformed covariance function with the horizontal plane yields an ellipse with one of its axes pointing toward direction r. To calculate a value on this function the two components along the axes have to be computed first. The evaluation of the prediction function through linear combinations is identical to the previously described method.

A different way of interpretation for affine transformed covariance functions is the following. One uses rotational symmetrical covariance functions and affine transforms the plane position of the reference points instead. Thus a rectangular grid can be changed into a square grid for instance. The prediction is then performed on the square grid of transformed reference points, the generated prediction functions are later transformed back.

Normalizing the Weight Function The purpose of normalizing the weight function can be explained with the aid of an example. If all reference points had a height of 1 one would generally expect a horizontal plane to be predicted. The prediction through linear combinations of both above mentioned covariance functions fulfills this expectation with very close approximation, unless the covariance function parameter is chosen too small in the case of the bell curve. A decrease of the predicted surface between the reference points is the result. The same effect occurs if the reference points have differing heights.

The decrease can be corrected through the following measures: One introduces a second right-hand side into the system of equations with value 1 (i.e. all reference heights = 1), now it can be recognized by how much the prediction surface departs from the horizontal plane. If the predicted value gotten from the reference values is divided by the predicted value gotten from reference heights =1 then the decrease effect disappears and a regularly rounded surface is generated, and if necessary a horizontal plane.

Application of Linear Prediction in SCOP

The Choice of the Covariance Function The covariance functions *Bell Curve* and *straight Line* yield different results. This becomes evident already by looking at the graphs of the functions. Considering the rotational surfaces formed by the covariance functions, the bell curve has a horizontal tangential plane and a continuous derivative at the rotation axis, while the straight line shows a peak in this point. Since the bell curve is continuous and has a continuous derivative at any point these attributes can also be found in the prediction function. The surface thus formed will be smoothly curved. The use of the straight line results in more or less strongly prominent peaks in the prediction function at the reference points.

Figure 2 shows the characteristic results for a given set of reference points using the bell curve and the straight line. The results, shown in the example with a 1-dimensional distribution of reference points, can in principle be transferred onto the 2-dimensional distribution. For the straight line the prediction surface is evaluated nearly as if linearly interpolated from the neighboring points, whereas the bell curve uses further reference points to determine the curvature of the surface.

With the knowledge of these properties the use of the straight line or the bell curve as covariance function can be decided. If the surface should be smooth – and random errors filtered – the bell curve should be used. If the surface on the other hand shows a lot of irregularities the rigorous fit into the reference values tends to yield a swinging-out of the surface if the bell curve is used, which although is avoided in SCOP by automatically activated filtering. The surface is too inert to follow the movement of the reference values and too large filter values are the result.

If the straight line is used the surface will generate small peaks at every reference point and thus is bent. The reference points can now be rigorously fit into the surface even with very irregular structures.

The user has the choice which covariance function should be used. Experience has shown that the following solution represents a good compromise. If a CU does not contain break lines the bell curve is used. The presence of break lines in a CU indicates that irregular structures are present and thus the straight line should be applied as covariance function.

This compromise is the *default*; in SCOP++, it corresponds to radio button *Adapting* in group */Covariance Function*.

For smoothing and filtering random errors with given mean filter values, the bell curve has to be used. It seems absurd to use the straight line for smoothing because the straight line results in the generation of peaks, although using the straight line and changing the main diagonal terms for the determination of the unknown cluster parameters will lead to a smoothing effect in the sense of decrease the first and second derivatives.

The Determination of the Covariance Function Parameter If the bell curve is used as the covariance function for the linear prediction it is necessary to choose a meaningful covariance function parameter. The prediction result for the straight line on the other hand is hardly influenced by its covariance function parameter.

The amount of movement allowed by the prediction function when using the bell curve depends on the magnitude of the covariance function parameter. In other words the parameter determines how closely the prediction function can follow the reference values. The automatically activated surface smoothing eliminates a swinging-out of the prediction function if the covariance function parameter was chosen too large and increases filtering and smoothing. Equally the choice of a too small covariance function parameter would lead to a decrease of the prediction function, if normalization is not used. For very irregularly distributed reference points the choice of the covariance function parameter is more difficult and a different parameter should be used for each covariance function, which in turn would lead to too much computing time needed (asymmetric system of equations).

As a conclusion of the above thoughts the determination of the covariance function parameters by SCOP is described. For each reference point the distance to its closest neighbor is calculated. The average of these distances is taken. In addition it is checked if gaps are existing between the reference points. If for a specific grid point all reference points are far away normalization is used (see 18.2.2). The covariance function parameter can not drop below a minimum value determined from the grid step.

Figure 3 shows the dependence of the prediction function from the covariance function parameter for the case of the bell curve.

Not: the user cannot influence parameter m of the covariance function directly (see 18.2.2)

Control of Surface Smoothing Surface smoothing with filtering has to fulfill three tasks. First, to avoid that, owing to two reference points with nearly identical XY coordinates but different heights, the prediction function swings out. Second, to provide the user with the possibility of a qualified surface smoothing. And third, to filter accidental errors in reference data.

In section 18.2.2 it was noted that by changing the main diagonal elements certain reference points would not fit the prediction function anymore in favor of a smoother surface. The degree of smoothing is amplified if, with fixed changes to the main diagonal elements, the covariance function parameter was increased.

To control smoothing in SCOP mean desired filter values are input into the program one for each point class. The four point classes distinguish bulk points, spot heights, form

line points and break line points (*Stochastic Filtering*, column *Aimed at: bulk points, spot heights, form lines, break lines*).

Since the reference values are centered on a horizontal plane or a tilted plane their variance can be calculated which allows certain conclusions about their size to be drawn. The mean filter values desired for the bulk points can be put into relationship with the variance which leads to a control over the covariance function parameter. If the mean filter value is small in comparison to the variance the covariance function parameter, which was calculated from the distances of the closest neighbors to each point, is kept constant; while the covariance function parameter is increased if the mean filter value becomes close to the variance.

After checking the covariance function parameter whether to keep it constant or to adjust it by a certain amount, the value to be added to each main diagonal element must be computed. For this a distance model is established. This distance model shows for each point the distance to the closest reference point belonging to the same point class or a point class having a larger desired mean filter value. If for instance a spot height has the smallest associated desired mean filter value and this point is the only spot height within the CU of interest it will be associated with the distance infinity.

The value added to the main diagonal elements is calculated from the ratio between the determined distance and the covariance function parameter. If the ratio is bigger than 1 nothing is added to the main diagonal element which means that this point will fit exactly. In the above example the spot height will fit exactly. The consideration of the different desired mean filter values for the evaluation of distances follows a priority principle. Even small differences in the desired filter values decide over filtering of one point class and fitting of another point class.

While the mean filter values for the bulk points are considered during the resizing of the covariance function parameters, the given mean desired filter values of the other point classes are put into relationship with that of the bulk points. The result is a factor for each point class with which the value to be added to the main diagonal elements is multiplied. This factor results either directly from the ratio of the mean desired filter values given for each of the four classes or if there are ratios bigger than 10 from a distribution of the factors in a range from 1 to 10 corresponding to their size.

Summarizing filtering and surface smoothing the following can be stated:

- The desired mean filter value of bulk points controls the degree of smoothing
- *Even small differences in the desired mean filter values of the four point classes decide which point class is preferred in fitting onto the predicted surface.*
- *Filter values can be chosen very small (e.g. millimeter), so that filtering only occurs if points are close together.*

From a statistical point of view, the desired mean filter values are the standard deviation of Z co-ordinates at the reference points (Kraus (2000)). The *a posteriori* filter value (see in the protocol file of DTM interpolation) should be close to the *a priori* value specified in group *Smoothing / Filtering / Blunder Detection*.

Checking of the Filter Values Surface smoothing as described above may result in too strong smoothing if the point distribution and the structures are very irregular. The evaluated prediction function departs too much from measured values.

The user has the possibility to enter a maximum allowed filter value for each point class (*Smoothing / Filtering / Blunder Detection, Switch to Straight from ...*). If this is exceeded it is assumed that the terrain is very irregular and the straight line is used as covariance function. The maximum allowed filter value should be chosen rather large to avoid switching to the straight line too often, because in overlapping surveying areas for instance no more smoothing would be done.

The problem that, for instance in overlapping areas, larger maximum errors, not acceptable in the central areas, will occur remains.

The user is given the possibility to demand an output of actual filter values exceeding a given value, with the point's XY co-ordinates listed. He can check if there really are discrepancies in the data or if the smoothing / filtering is too strong (*Smoothing / Filtering / Blunder Detection, Protocol for ....*

Affine Filtering, Filtering Scanning Errors It is possible to use affine transformed covariance functions in SCOP, although this is limited to the bell curve. The main purpose for this is to eliminate scanning errors, although it may also be applied successfully on points distributed at random with different densities in two mutually perpendicular directions.

The direction, the distance between profiles and the average point distance along the profiles can be specified by the user (checkbox *Apply Affine Filtering*, *Direction, Profile Distance, Point Distance*). If one of the values is unknown or if one value changes within the DTM, it may be determined automatically for each CU by the program. It is assumed however that for the automatic determination the points are read in in the sequence in which they have been scanned.

The use of affine transformed covariance functions, here always referred to as elimination of scanning errors, is possible for any distribution of points and there need not be any relationship between the input values and the point distribution. If the the points are input in the above described manner it will generally lead to the desired result.

As described in section 18.2.2 the elimination of scanning errors may be interpreted as an affine transformation of the reference points. If the input values reflect the reality then the affine transformation leads to a grid with side length 1 in the direction of the profiles and 0.5 perpendicular to the profiles.

The values added to the main diagonal elements is derived from the ratios of the mean desired filter values. The covariance function parameter for the bell curve is 1.5. The rectangular distribution results in the ability of the prediction function to follow the profiles, but it is too inert to follow in a perpendicular direction to the profiles. Thus the surface is smoothed perpendicular to the direction of the profiles.

An increase of the profile distance and the point distance leads to a simulated point distribution with smaller side lengths, so that smoothing acts even stronger. Sinusoidal contour lines running perpendicular to the profiles are smoothed if the profile distance is increased, while an increase of the average point distance leads to more equally spaced contour lines.

If the terrain was surveyed by parallel profiles, and the profile distance is different to the average point distance along the profiles, certain effects can be eliminated from the

prediction function, if the affine transformation of the covariance function is used. Often prediction surfaces are generated in the above situation leading to closed contour lines between neighboring profiles or closed contour lines around single points on a profile. If the direction of the profiles, the point distance along the profiles and half the profile distance are introduced a square grid is simulated as point distribution, and the closed contour lines disappear. The often undesirable closed contour lines result from a flattening of the interpolated profiles.

18.3. Remarks Concerning Data Acquisition

The program system SCOP works with almost all types of data from topographic terrain surveys. The above description of the algorithms used shows however that certain rules have to be accounted for during the survey to get optimal results.

18.3.1. Point Distribution

A computing unit including its overlap area may include a maximum of 800 points. If in a square made up of 3*3 CU's there are no reference points a gap is generated in the DTM. Thus the point density may vary within these limits in order to avoid too many points in a CU and gaps in the DTM.

The use of variable overlap guarantees that prediction functions of neighboring CU's join up even with irregular point densities.

18.3.2. Break Lines

Break lines represent surface edges. They influence the generation of interpolation areas as well as the use of the covariance function. Since break lines are joined up by the program an exact joining of break lines during the survey is not necessary, i.e. break lines need only join up approximately. Here it is advantageous to end lines slightly earlier and let the program do the joining up. Thus intersecting break lines, most often having different heights which are rigorously taken over into the DTM, can be avoided.

Break lines may be densified by the program to enable widely spaced acquisitions. It must be possible however to interpret them as polygons in plane position as well as in their height.

18.3.3. Border Lines

The running of border lines must be logically correct. Border lines are only joined if their co-ordinates are exactly identical. Logically correct implies for instance that, if a closed border line excludes an area inside, a second border line lying inside the excluded area must then again activate an included area inside its limits. Intersecting border lines are logically wrong.

It is possible to delimit the total surveyed area by a border line and thus exclude all external parts.

Free ending border lines have to end at least two CU's outside the last CU of the DTM, thus parts on the edges may be excluded.

18.3.4. Spot Heights

Spot heights are measured at selected points of the terrain for instance depressions or peaks. Since bulk points are often surveyed with far lesser accuracies in comparison to spot heights, the filter values of spot heights should be chosen rather small to avoid the reversal of a high point to a low point or vice versa owing to the error in height of a nearby bulk point.

18.3.5. Form Lines

Similar to break lines, form lines should be surveyed such that a spatial polygon going through the points represents the line with satisfying accuracy.

If this does not already lead to a point distance smaller than that of the bulk points, it is recommended to call the automatic form line densification. This guarantees that the prediction surface represents the form lines correctly.

18.3.6. Overlapping Surveying Areas

For the optimal acquisition of spot heights, form lines, break lines and border lines the surveying units should be separated from each other to avoid double surveys of the same areas. Without knowledge of the terrain surface one often can not distinguish which lines should be identical and thus be connected with each other or perhaps should be averaged. The same applies to spot heights.

19. DTM Computation Using Fast Interpolation

19.1. The Task

The task is to develop and realize a method of interpolation which minimizes computing times. An exact description of the terrain is neglected in favor of the reduced amount of calculation.

Given are, as in the case of linear prediction, the recorded points subdivided into bulk points, spot heights, form line points, break points and border line points. The structure of the DTM is to be kept identical to that of the prediction so that break lines, form lines and border lines are integrated into the DTM. The heights of the grid points and the grid intersection points are to be computed by a simple method.

For a fast interpolation it is impossible, in contrast to the linear prediction, to consider all points within a CU and its overlap area for the evaluation of grid points and grid section points. Only directly neighboring points can be taken into account. The consideration of break lines in a simple interpolation also leads to a considerable increase in the effectiveness of the method. Since only direct neighbors are considered for the evaluation of an interpolated height discontinuities in the description of the terrain surface arise as soon as a point is discarded or if another one is added.

The fast interpolation will result mainly in a loss of accuracy if the point distribution is irregular and if the terrain is difficult, or if the measurements have a bigger random error.

The loss of accuracy has a lesser effect in the heights than on the computed surface inclination and curvature. The contour line image derived from the fast interpolation is often slightly more irregular.

The method of fast interpolation (*Moving Plances*) used in SCOP is described in section 19.2. It is possible to interpolate the DTM by the weighted arithmetic mean of the measured heights (*moving average*), or according to the method of *moving tilted plane*.

19.2. The Solution

The evaluation of the DTM with fast interpolation follows a pattern similar to the prediction method. The computation is also done on a computing unit basis, starting with the lowest strip from left to right. The evaluation of a single height is done by subdividing its neighborhood into 4 or 8 sectors, and one point out of each sector is chosen under consideration of break lines as regional boundaries. The height is then calculated as the

weighted mean of the heights of the surrounding points, or a best fitting tilted plane is calculated through the 4 or 8 points by the method of least squares with its height in the grid point corresponding to the interpolated height.

The computation of one CU can be subdivided into the following steps, the most importants of which are more closely discussed in the next section. As in the case of prediction the break lines, form lines and border lines as well as spot heights are inserted into the DTM.

a) Preparations for the Computation of the DTM
   - Reading-in of all points lying in the CU and its overlap area
   - Insertion of break lines, form lines and border lines into the DTM grid
   - Break line densification, form line densification

b) Computation of the Heights of the Grid Points and the Grid Sections for each grid point
   - Subdivision of the neighborhood into 4 or 8 sectors
   - Choice of one reference point from each sector
   - Computation of the height
     - by weighted arithmetic mean
     - moving tilted plane

     for the grid sections
   - Linear interpolation between the two given surveyed line points

19.2.1. Definitions and Descriptors

The following descriptors are of fundamental importance and are explained in the following sections.

- Subdivision into sectors
- Weighted arithmetic mean
- Moving tilted plane

19.2.2. Algorithms

The input of surveyed points and the connection of break lines and border lines is done by an identical scheme as in the case of linear prediction. In addition the following new algorithms are introduced.

Subdivision into Sectors, Choice of Reference Points

For each grid point the neighborhood is subdivided into either 4 or 8 sectors. The boundaries of the sectors are straight lines running through the DTM grid point to be interpolated. If 4 sectors are used the two angular bisectors to the co-ordinate lines are taken. If 8 sectors are used the two co-ordinate lines are taken in addition to the 4 sector case.

Out of each sector the closest reference point is chosen so that generally 4 or 8 points are available. The subdivision into 4 or 8 sectors guarantees that the chosen points cover the area in the neighborhood. If no point can be found in one sector the evaluation of the height has correspondingly less points available.

If it becomes impossible to uniquely define a tilted plane through the chosen points the arithmetic mean is used.

If there are break lines in the CU their intersections with the sector boundaries are linearly interpolated between the break line points and these are treated as additional reference points. The effect of taking reference points beyond a break line into account is thus avoided.

The Computation of Grid Heights by Arithmetic Means

If n reference points are given the interpolated value can be calculated by

$$z = p1 * s1 + p2 * s2 + \cdots + pn * sn$$

with $s1 \ldots sn =$ reference value
$pj =$ weight of reference value$j$

The weights are chosen according to

$$p = 1/(d * d)$$

where d is the distance from the reference point to the point to be interpolated.

The Tilted Plane

The equation of a tilted plane is

$$z = a[1] * x + a[2] * y + a[3]$$

If n reference points are given then the unknowns a are calculated from a parametric least squares adjustment. The observation equation for each reference point j with co-ordinates u[j],v[j],s[j] is

$$v[j] = a[1] * u[j] + a[2] * v[j] + a[3] - s[j]$$

The weight corresponding to each observation results from the distance d between the reference point and the point to be interpolated according to

$$p = 1/(d * d).$$

If the point to be interpolated corresponds to a reference point it will get its height since here the corresponding weight would be infinity.

20. DTM Computation Using Hierarchic Robust Interpolation

Depending on the data acquisition method gross errors (blunders) can occur in the data. They can be caused by human failure, by miss-matches in correlation programs, by multi-path effects or reflections on vegetation and buildings in airborne laser scanner data, or by above-ground obstacles in sonic depth finder data. These points can be either above or below the ground. For airborne laser scanner data the gross errors are typically above the terrain, resulting in a skew distribution of gross errors. The technique of *hierarchic robust interpolation* allows to eliminate those blunders automatically.

20.1. The robust interpolation

In this section the theoretical background for the robust interpolation will be presented. It is the core of the *hierarchic* robust interpolation which will be described in the following sections. To make the description demonstrative it will be explained for laser scanner data. Of course, the method works similar for data with a different distribution of gross errors.

The algorithm is based on linear prediction with individual accuracies for each measurement. It works iteratively. In the first step all points are used to estimate the covariance function of the terrain. The surface is computed with equal weights for all points, or to be more precise, for all z-measurements z_i. This surface runs in an averaging way between ground points and vegetation points. The ground points are more likely to

- be below the averaging surface,
- have negative filter values,
- have positive residuals.

Whereas the vegetation points are more likely to

- be above or on the averaging surface,
- have positive filter values,
- have negative residuals.

The filter values f_i are defined as the negative of the residuals r_i. If h_i is the terrain height interpolated at point i, than it holds:

$$\begin{aligned} h_i &= z_i + r_i \\ z_i &= h_i + f_i \\ -r_i &= f_i \end{aligned}$$

These filter values f_i are used to compute weights p_i for each measurement (z_i). The following weight function is appropriate for airborne laser scanner data (see Figure 20.1):

$$p_i = \begin{cases} 1 & f_i \leq g \\ \dfrac{1}{1+(a(f_i-g))^b} & g < f_i \leq t \\ 0 & t < f_i \end{cases} \tag{20.1}$$

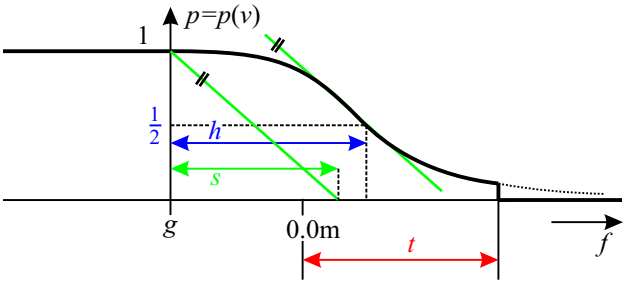


Figure 20.1.: Weight function: At h above g the weight function has a half of its maximum value, h is the half–width value. The value s determines the steepness of the weight function at this point. At t in the right tail the weight function is cut off.

This is the standard weight function of robust estimation, adapted to the special needs of robust filtering. It is shifted by the eccentricity g, and the left branch is identical to one. For laser scanner data g is negative. The value for g can be computed with a histogram of the residuals. The interpretation of the weight function is more obvious if the half–width value h and the slope s at this point are used than if the parameters a and b are used.

$$\begin{aligned} a &= \frac{1}{h} \\ b &= 4\frac{h}{s} \end{aligned} \tag{20.2}$$

Now the weights p_i can be used to compute variances for each measurement (weights are indirect proportional to variances). These variances can be considered in the next iteration step, i. e. the next interpolation. It is necessary to cut off the weight function at a certain value (see Figure 20.1). Setting the weight p_i of a measurement identical to zero, means that this point is not used for the interpolation of the surface.

The formulas for the linear prediction, which have already been presented in section 18.2.2 have to be altered a little bit to allow for the consideration of individual accuracies. Given are the points $\mathsf{P_i}$ with heights z_i, which have been reduced by subtracting a trend surface. We want to find a surface – height z at position P – which interpolates or approximates the given data points.

$$C(\mathsf{P_iP_k}) = C(0)\mathrm{e}^{-\left(\frac{\mathsf{P_iP_k}}{c}\right)^2} \tag{20.3}$$

$$z = \mathbf{c}^T\mathbf{C}^{-1}\mathbf{z} \tag{20.4}$$

$$\mathbf{c} = \left(C(\mathsf{PP_1}), C(\mathsf{PP_2}), \ldots C(\mathsf{PP_n})\right)^T$$

$$\begin{aligned}\mathbf{C} &= \begin{pmatrix} V_{zzp_1} & C(\mathsf{P_1P_2}) & \dots & C(\mathsf{P_1P_n}) \\ & V_{zzp_2} & & C(\mathsf{P_2P_n}) \\ & & & \vdots \\ symm. & & & V_{zzp_n} \end{pmatrix} \\ \mathbf{z} &= (z_1, z_2, \dots z_n)^T \end{aligned}$$

Here $\mathsf{P_1P_2}$ denotes the distance between the points $\mathsf{P_1}$ and $\mathsf{P_2}$ in the ground plan. The V_{zzp_i} are obtained from:

$$V_{zzp_i} = \sigma_i^2 + C(0) \tag{20.5}$$

$$\sigma_i^2 = \frac{\sigma_0^2}{p_i} \tag{20.6}$$

These formulas are not only applicable for filtering airborne laser scanner data. If a different distribution of gross errors has to be handled, this results in a different setting of the weight function. The general form of the weight function in SCOP++ is:

$$p_i = \begin{cases} 0 & & & f_i & \leq & t_- \\ \dfrac{1}{1+(a_-(g-f_i))^{b_-}} & t_- & < & f_i & \leq & g \\ \dfrac{1}{1+(a_+(f_i-g))^{b_+}} & g & < & f_i & \leq & t_+ \\ 0 & t_+ & < & f_i & & \end{cases} \tag{20.7}$$

The left and the right branch of the weight function can be set independently from each other. The values a_+ and b_+ determine the right branch, a_- and b_- are responsible for the left branch. Of course, these values can be computed from half–weight values $(h_+, h_-)$and slants (s_+, s_-) with Eq. 20.2. As mentioned above, g can be determined automatically. There are three choices regarding the determination of g.

1. $g = 0$, this sets g to zero. This is applicable, if no special distribution of gross errors is known before the interpolation.
2. $g \leq 0$, this triggers the automatic determination of g with the restriction, that g is not larger than zero. This is the applicable for airborne laser scanner data.
3. $g \geq 0$, this triggers the automatic determination of g with the restriction, that g is not smaller than zero.

If the value of g shall be determined automatically, a histogram of the residuals is computed and analyzed. The methods are:

- One method of finding g is based on an estimation of the penetration rate of the laser scanner data. If the penetration rate is estimated to be r%, then g is set to the $\frac{r}{2}$%-quantile.
- A different method computes (theoretically) for each real value x the standard deviation of the residuals, which are smaller than x: $s_x = \sqrt{\sum(v_i - x)^2/n_x}$, with n_x the number of residuals $v_i \leq x$. In general, s_x is continually growing with x. For a predefined accuracy σ, g can be chosen such that $s_g = \sigma$. For airborne laser scanner data σ is the expected accuracy of the ground points.

- Finally, some threshold values are used to insure that g is not too small or too large.

The choice of method can be influenced by specifying a penetration rate. If no penetration rate is specified, this method will not be used. The other method will always be used, if the automatic determination of g is specified.

20.2. The hierarchic approach

As mentioned above, the robust interpolation is embedded within an hierarchic approach. It works in a coarse to fine strategy, adding more and more (terrain) detail in each step.

The approach is comparable to a hierarchical setup using image pyramids, in this case data pyramids. The structure of these pyramids is regular (as in image processing) and typically two or three levels are sufficient. However, in comparison to image analysis, the reduction function operates on point data, not on pixels. The method proceeds as follows:

1. Create the data pyramid (starting from the the lowest resolution)
2. filter the data robustly and generate a DTM,
3. compare the DTM to the data of higher resolution and take points within a certain interval.

This process (steps 1,2 and 3) is repeated for each level.

There are many possibilities to generate a coarser resolution (higher levels) of the given data (lowest level: level 0 is the original data), the most straightforward techniques are to place a square grid over the data and compute for each cell one new point (regular pyramid). There are different methods to compute this point which are applicable:

- Take the mean point in each cell.
- Take the lowest point in each cell.
- Take the point closest to the center in each cell.
- Take the highest point in each cell.

Only the first of these reduction functions is linear (fast to compute), whereas the others are nonlinear. As the point density itself is diminished in this step we call it a *thin out*. The second method speeds up the filtering for airborne laser scanner data, but there is the danger of taking a point which is a negative blunder (long ranges). The third method provides a more regular structure of the generated point set. Finally, the fourth method is applicable if a tree canopy model should be computed. The choice of method can also be made depending on the kind of data (measurement method, blunder rate, city or wooded area, ...). A thin out of the data has two advantages:

1. The robust filtering is accelerated, because a smaller data set is handled.
2. The regular structure of airborne laser scanner data in built up areas (long sequences of points on the ground, on the roof, resp.) is broken apart.
3. By applying a ´normal' DTM computation after the thin out, an overview can be generated very fast.

With the coarser data set a filtering is performed and a DTM is generated. Of course, embankments, break lines and other features are represented less distinctively than in the original data, but this is due to the coarser level of data, not due to the filtering. Therefore, for performing the filtering on the next lower level, the original point data (or the data of the corresponding pyramid level) and the DTM of the coarser level have to be combined to generate the data for the lower (finer) level. The original points and their distances to the DTM at the current resolution are analyzed. If they are within a certain interval they are accepted, otherwise they are rejected (sorted out). Thus, this step is called *sort out*.

$$\begin{aligned} \text{Given set of points:} \quad & \mathsf{P_i} = (x_i, y_i, z_i) \\ \text{Computed terrain model:} \quad & h(x, y) \\ \text{Selected points:} \quad & t_- \leq z_i - h(x, y) \leq t_+ \end{aligned}$$

Of course, the values t_-, t_+ can, but need not be the same as in equation 20.7 For laser scanner data the lower bound of this interval is negative (e.g. -1m) and the upper bound is positive (e.g. 2m). With this data set the next filter step can be performed. The choice of the intervals is not as critical as one might expect first, which is due to the filter step that is performed afterwards. Some blunders may be included in the data set because they are relatively small, but they can be filtered in the next step.

21. Input of a Measured or Pre-computed Grid

21.1. The Task

It should be possible to enter a directly measured DTM grid, where additionally measured break lines, form lines, border lines and spot heights must be added. A scanning error present in the measured grid should be eliminated easily.

As in the cases of linear prediction and fast interpolation the input data consists of bulk points, spot heights, break lines, form lines and border lines. The structure of the DTM is to be kept as in the other methods, so that break lines, form lines and border lines are inserted into the DTM.

Since here the plane positions of the bulk points are identical to those of the grid points the height of each grid point is taken over from its corresponding bulk point. For the elimination of scanning errors the bulk point lies in the middle between two grid points. The grid height is taken as the average height of the two neighboring bulk points. If it is impossible to associate the bulk points with the grid points however one has to interpolate.

21.2. The Solution

As in the cases of linear prediction and fast interpolation, the external structure of the evaluation of the DTM is similar in this case. Here too the computation is done on a computing unit basis starting with the lowest strip from left to right. The following step is to store the bulk points into the DTM and for the elimination of scanning errors the averaging of neighboring profiles.

If storing of the bulk points is impossible owing to lack of data the CU will be interpolated either by prediction or by fast interpolation.

If the measured grid is variable (e.g. progressive sampling) the grid step is chosen such that the smallest interval in the CU is taken, missing grid points have to be interpolated.

As in the other cases break lines, form lines, border lines and spot heights are inserted into the DTM.

The following steps are executed for each CU:

a) Preparation of the DTM Computation
   - Reading-in all points lying within the CU and its overlapping area
   - Insertion of break lines, form lines and border lines into the DTM grid

b) Computation of the Heights of the Grid Points and the Grid Sections for each grid point

- Storing of the heights of the bulk points
- Averaging during elimination of scanning errors
- If necessary interpolation of grid points

for each grid section

- Linear interpolation between the given break line points, form line points and border line points

22. The Derivation of Contour Lines

The contour lines are derived from the SCOP DTM. The information contained in the SCOP DTM are the grid heights, the grid sections, the points of the break lines, form lines and border lines and the spot heights. These informations have to be considered during the derivation of contour lines.

The most important steps to get from the DTM to the contour lines result from the structure of the DTM.

- Derivation of contour lines within a CU
- Linking of contour line sections gotten from the CU's
- Preparation of the contour lines for graphical output

The program SCOP.ISOLINES contains further a derivation of intermediate contour lines (change of contour interval).

22.1. Determination of Contour Lines within one CU

Firstly the contour lines are worked through separately for each CU which leads to contour line pieces. The sequence used here is from bottom to top. It is first checked if the current contour line enters the CU at the edge of the CU. The contour line is then followed from one grid square to the next. It is then checked if there are additional closed contour lines within the CU. Here again the contour lines are followed grid mesh by grid mesh.

The position of a contour line within a grid mesh depends on the grid heights and on the existence of break lines, form lines, border lines and spot heights.

22.1.1. Determination of Contour Lines from Grid Heights

If within a grid mesh there are only the grid heights given, and there are no break lines, form lines, border lines or spot heights present, then the contour line is interpolated linearly along the grid lines. The intersection of the grid lines with a horizontal plane at the height of interest results in either 2 or 4 intersection points. In the case of 2 points they are joined to describe the line.

In the case of 4 points the question is raised which two points are to be joined. To answer the question grid points lying further away have to be considered. From the 4 points and the 8 additional points the height of the center of the grid mesh is interpolated. The logically correct position of the contour line piece can be deduced by looking at the grid diagonals, since the connections between the 4 points are either closed or opened by the diagonals.

22.1.2. Consideration of Spot Heights

For the derivation of contour lines spot heights have to be taken into account, since otherwise discrepancies will result between the image of the contour lines and the spot heights.

If a spot height exists within a grid mesh the connections between the spot height and the grid points are calculated and checked whether additional intersections of the horizontal plane at the height of interest with the connections exist. Generated intersections are taken over into the contour line. It is also possible that new closed contour lines are generated around that spot height.

22.1.3. Consideration of Cuts between the Grid and Break Lines

During the search for intersections of the contour interval with the grid existing intersections of break lines with the grid are considered. Thus a new contour line point may result after each intersection of the grid with a break line.

Beside the intersections along the grid lines the search is extended onto the break lines themselves. If no intersection occurs along a break line the derived points have to be connected such as not to cut the break lines. If there are cuts they are taken over into the contour line. At the cuts of contours and break lines the arriving contour is finished. The following contour part is stored as a new contour. This allows contour lines to bend off at a break line even if curved lines are used for the contour plot.

22.1.4. Consideration of Cuts between the Grid and Form Lines

The grid intersections of form lines are also included into the contour line determination similar to break lines. The difference here is, in opposition to break lines, that the contour lines are not interrupted, because at a form line the contour line should not bend off.

22.1.5. Consideration of Cuts between the Grid and Border Lines

It is checked if a contour line intersects a border line. If an intersection occurs the continuation of the contour line is either suppressed or initialized dependent on whether the border line represents the beginning or the end of an excluded area.

22.1.6. Consideration of Break Line, Form Line and Border Line Points

The sole consideration of grid intersections of break lines, form lines and border lines may, with large grid size, not yield the desired accurate results. The points used to define the break lines, form lines or border lines have to be considered as well. If for instance the break line points were not considered, it could happen that at a point where the break line changes direction the contour line intersects the break line even if the heights of the break line do not indicate an intersection.

The break line, form line and border line points are connected to the corner points of a grid mesh by artificial lines. If two lines end or begin with identical plan co-ordinates

their ends are joined. If two lines intersect their point of intersection is interpolated and linked.

It is checked if intersections with the contour lines occur along the artificial lines, too. These intersections are also considered for the path of the contour line.

While at embankments the contour lines follow the edges quite well complications may arise if too many points fall within a single grid mesh and in addition the heights of the break lines do not agree with each other. In this case small closed contour lines can be be generated.

The linking of break lines, form lines and border lines can for this reason also be switched off.

22.2. The Concatenation of Contour Lines Pieces

The contour line pieces evaluated in the individual CU's are put together. Thus continuous contour lines are generated, which are either closed or open and thus end at the map edges, the border of the surveyed area or at border lines.

22.3. Graphical Representation of Contour Lines

Once the paths of the contour lines are completed, they can be prepared for the output to a specified plotter. The user is given various possibilities with respect to map representation especially with respect to map frames, line types and contour labeling.

The contour lines may be labeled in different intervals and with varying distance, small closed lines can be suppressed and depressions can be represented by arrows. Different line types are also at the user's disposal.

For the graphical output it is useful to rearrange the point sequence defining the contour lines. The evaluation of the contour lines results in a sequence of grid intersection points and some additional points. On a plotter a point sequence with equally spaced points is more useful. Here one has to take into account if the plotter is only able to connect points by straight lines or if it possesses powerful curve interpolation algorithms which avoid stopping the plotter at every new point.

22.4. Intersecting Contour Lines

Touching or intersecting contour lines may arise in some special cases.

If a grid point has the same height as a contour line and if at the same time this point defines a saddle point the four intersections along the grid lines become a single point. This leads to two touching contour lines in this point. Mathematically this is correct, in topographic maps however this effect is not desirable.

Contour lines are derived as polygons; interpolating curves from them to be plotted may lead to line intersections. Intersections may especially occur in case of very close neighboring contours and in case of using less suitable curve functions. Intersections can often be avoided by using a smaller DTM grid step.

Intersecting break lines with different heights in the intersection point may also lead to intersecting contour lines. The linking of break line points can sort out this problem to a certain degree if the connections to be formed are not too complex. In general the description of the terrain surface is not unique if there are intersecting break lines.

22.5. Intermediate Contour Lines

The problem of drawing intermediate contour lines in cases when contour lines are far apart can hardly be solved with acceptable programming and computation efforts in a program. For this reason a different decision criterion was chosen in SCOP.

Whether or not an intermediate contour line is computed and drawn is decided on the criterion of the slope of the surface. The desired contour interval can be subdivided into two further intervals and a value for the slope of the terrain has to be given by the user as the deciding factor. In each CU it is checked whether a grid point has a slope less than the given limiting value in order to decide if the evaluation of an intermediate contour line is meaningful or not. As with the other contour lines the pieces of the intermediate lines are connected. To avoid very short intermediate lines and to join up pieces ending very close to each other the user can decide upon the minimum length of such a line and upon the distance over which an intermediate line is to be continued although the points in between have a larger slope.

The “conventional” result described above is generated if the terrain is regular and flat. The intermediate lines will then lie regularly spaced between the other lines. Very often flat terrains are slightly undulating. This may lead to the fact that the slopes are larger than expected by looking at the distance between contour lines and thus intermediate lines are not evaluated. If intermediate lines are present they will represent the small form contained in the terrain and thus are not necessarily similar in shape to their neighboring contours.

It is also possible that in steep areas a narrow horizontal strip causes the evaluation of intermediate contours.

To switch off small intermediate pieces and to join pieces ending very close to each other a minimum line length and a minimum gap length can be entered.

23. Literature

23.1. General Basic Literature

A. Flotron and O. Kölbl. 2000. Precision Terrain Models for Civil Engineering. *In: European Organization for Experimental Photogrammetric Research*.

D. Heitzinger and H. Kager. 1998. High Quality DTMs from Contourlines by knowledge-based classification of problem regions. *Pages 230–237 of: Proceedings of the International Symposium on: GIS Between Visions and Applications*, vol. 4.

D. Heitzinger and H. Kager. 1999. Hochwertige Geländemodelle aus Höhenlinien durch wissensbasierte Klassifikation von Problemgebieten. *Photogrammetrie-Fernerkundung-Geoinformation*, **1**, 29–40.

Kraus, K. 1997. *Photogrammetrie*. Vol. 1. Dümmler Verlag, Bonn.

Kraus, K. 2000a. *Photogrammetrie, Topographische Informationssysteme*. Vol. 3. Dümmler Verlag, Bonn.

Kraus, K. 2000b. *Photogrammetry*. Vol. 1. Dümmler Verlag, Bonn.

Ruedenauer, H. 1980. *Zur photogrammetrischen Erfassung von Geländedaten und deren digitaler Verarbeitung unter Berücksichtigung strassenbaulicher Forderungen*. Wissenschaftliche Arbeiten der Fachrichtung Vermessungswesen 101. Universität Hannover.

Schut, G. H. 1976. Review of interpolation methods for digital terrain models. *In: Proceedings of the XIIIth ISP-Kongress*. ISPRS, Helsinki. Invited Paper, Kommission III.

Torlegard, K. 1984. A comparative test of photogrammetrically sampled digital elevation models. *In: Proceedings of the XVth ISP-Kongress*. Invited Paper, Kommission III.

23.2. Literature Concerning Theory

E. Assmus and K. Kraus. 1974. Die Interpolation nach kleinsten Quadraten – Prädiktionswerte simulierter Beispiele und ihre Genauigkeit. *In: DGK, Reihe A*, vol. 76.

Kraus, K. 1972. Interpolation nach kleinsten Quadraten in der Photogrammetrie. *BuL*, **40**, 7–12.

Kraus, K. 1973. Ein allgemeines digitales Geländemodell - Theorie und Anwendungsmöglichkeiten. *Pages 225–251 of:* Ackermann, F. (ed), *Numerische Photogrammetrie*. Sammlung Wichmann, Neue Folge, vol. 5. Wichmann.

Kraus, K. 1984. *Photogrammetrie*. Vol. 2. Bonn: Dümmler-Verlag. Pages 231–322.

Kraus, K. 1998. Interpolation nach kleinsten Quadraten versus Krige-Schätzer. *Österreichische Zeitschrift fuer Vermessung und Geoinformation*, **86**, 45–48.

Kraus, K. 2000. *Photogrammetry - Advanced Methods and Applications*. Vol. 2. Dümmler Verlag, Bonn.

Wild, E. 1983. Die Prädiktion mit Gewichtsfunktionen und deren Anwendung zur Beschreibung von Geländeflächen bei topographischen Geländeaufnahmen. *In: DGK, Reihe C*, vol. 277. Dissertation an der Universität Stuttgart.

23.3. Literature Describing the SCOP Programs

SCOP++ Data Sheet

SCOP Product Information

Ackermann, F. 1985. Data processing aspects of digital elevation models. *In: Conference of Southern African Surveyors*.

Assmus, E. 1975. Extension of Stuttgart Contour Program to treating terrain break lines. *Pages 171–178 of: Proceedings of the symposium of the ISP, Commission III, Stuttgart 2.-6.9.1974*. DGK, Reihe B, vol. 214.

Ecker, R. 1991. Rastergraphische Visualisierungen mittels digitaler Geländemodelle. *In: Geowissenschaftliche Mitteilungen*, vol. 38. TU Wien.

Hochstöger, F. 1989. Ein Beitrag zur Anwendung und Visualisierung digitaler Geländemodelle. *In: Geowissenschaftliche Mitteilungen*, vol. 34. TU Wien.

Hochstöger, F. 1995. Die Ermittlung der topographischen Abschattung von GPS-Satelliten unter Verwendung eines digitalen Geländemodelles. *Österreichische Zeitschrift für Vermessung und Geoinformation*, **3**, 144–145.

Hochstöger, F. and Ecker, R. 1990. Application and visualization techniques for digital terrain models. *In: Symposium of ISPRS commission IV*. ISPRS, Tsukaba, Japan. Presented paper.

Köstli, A. and Siegle, M. 1986. Die SCOP-Datenstruktur zur Verschneidung und Korrektur von Geländemodellen. *Bildmessung und Luftbildwesen*, **3**.

Köstli, A. and Wild, E. 1984. Digital elevation model featuring varying grid size. *In: Proceedings of the XVth ISP-Kongress*. Presented Paper, Kommission III.

Kraus, K. 1989. *Retrospektives und Perspektives zum digitalen Geländemodell. Festschrift Friedrich Ackermann zum 60. Geburtstag*. Schriftenreihe des Instituts für Photogrammetrie, Heft 14. Universität Stuttgart.

Kraus, K. and Köstli, A. and Molnar, L. and Wild, E. 1982. *Digital elevation models: users' aspects*. Schriftenreihe des Instituts für Photogrammetrie, Heft 8. Universität Stuttgart.

Molnar, L. 1992. Principles for a new edition of the digital elevation modeling system SCOP. *Pages 962–968 of: International Archives for Photogrammetry and Remote Sensing, XXIX/B4*.

Molnar, L. and Assmus, E. and Köstli, A. and Wild, E. 1982. Digital Elevation Models: Informatics Aspects. *In: Symposium der ISPRS-Kommission III*.

Molnar, L. and Wintner, J. and Wöhrer, B. 1996. DTM System SCOP in a New Technological Generation. *Pages 569–574 of: International Archives for Photogrammetry and Remote Sensing, XXXI/B4*.

Rieger, W. 1991. Hydrologische Anwendungen des digitalen Geländemodelles. *In: Geowissenschaftliche Mitteilungen*, vol. 39b. TU Wien.

Stanger, W. 1973. Das Stuttgarter Höhenlinienprogramm, Beschreibung und Ergebnisse. *In:* Ackermann, F. (ed), *Numerische Photogrammetrie*. Sammlung Wichmann, Neue Folge, vol. 5. Wichmann.

23.4. Literature Concerning the Use of SCOP

Ackermann, F. 1980. *The accuracy of digital height models*. Schriftenreihe des Instituts für Photogrammetrie, Heft 6. Universität Stuttgart.

AdV der Länder der BRD - Arbeitskreis Topographie (ed). 1980. *Erprobung von Höhenlinieninterpolationsprogrammen - Schlussbericht*. Landesvermessungsamt Rheinland - Pfalz. Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV).

Dorffner, Mandlburger, Molnar, Wintner, Wöhrer. 1999. Geländemodelltechnologien - Forschung und Weiterentwicklung am I.P.F. *Pages 31–44 of: X. Internationale Geodätische Woche in Obergurgl*, vol. 18. Institut für Geodäsie der Universität Innsbruck. Presented paper, Kommission III, ISPRS-Kongress.

Ecker, R. 1999. (Homogenisierung digitaler Geländemodelle unterschiedlicher Genauigkeit mittels linearer Prädiktion und robuster Schätzung). *In: Proceedings des Symposiums für Angewandte Geographische Informationsverarbeitung*. AGIT.

INPHO GmbH. 1991. *Gutachten zur Erstellung von Erosionsgefährdungskarten*. Unveröffentlichter Bericht, erstellt im Auftrag des Landesamts für Flurbereinigung, Baden-Württemberg.

K. Kraus, E. Hynst, P. Belada, T. Reiter. 1997. Topographische Daten in bewaldeten Gebieten – Ein Pilotprojekt mit Laser- Scanner-Daten. *Österreichische Zeitschrift für Vermessung und Geoinformation*, **85/Heft 3**, 174–181.

Kiefer, L. 1986. Herstellung und Anwendung von automatisch erzeugten Gefällstufenkarten in der Flurbereinigungsverwaltung Baden-Württemberg. *Zeitschrift für Kulturtechnik und Flurbereinigung*, **27**, 210–217.

Kraus, K. and Jansa, J. and Kalliany, R. 1988. Visualisierungstechniken in der Photogrammetrie und Fernerkundung. *In:* Barth, W. (ed), *Visualisierungstechniken und Algorithmen*. Informatik-Fachberichte, vol. 182. Berlin Heidelberg: Springer-Verlag.

Rickenbacher, M. 1998. Das neue Panorama vom Bantiger. *Vermessung/Photogrammetrie/Kulturtechnik*, **3**, 114–118.

Schilcher, M. 1977. *A comparison of the accuracy of several contour plots of the Söhnstetten test field*. Schriftenreihe des Instituts für Photogrammetrie, Heft 4. Universität Stuttgart.

Schwartz, W. 1989. *Das digitale Geländemodell SCOP - vielfältige Einsatzmöglichkeiten im ÖbVI-Büro*. Tech. rept. 3. BDVI-Forum.

Sigle, M. 1984. *Ein digitales Geländemodell für das Land Baden-Württemberg*. Schriftenreihe des Instituts für Photogrammetrie, Heft 9. Universität Stuttgart.

Sigle, M. 1989. *Der Aufbau landesweiter digitaler Geländemodelle am Beispiel des DGM Baden-Württemberg. Festschrift Friedrich Ackermann zum 60. Geburtstag*. Schriftenreihe des Instituts für Photogrammetrie, Heft 14. Universität Stuttgart.

Stanger, W. 1982. Ein digitales Geländemodell und einige Anwendungsmöglichkeiten im Bereich der Flurbereinigung. *In: DGK, Reihe C*, vol. 273. Dissertation an der Universität Stuttgart.

Stanger, W. 1989. *Rechnerunterstützte Durchführung und Auswertung der Wertermittlung im Flurbereinigungsverfahren*. Schriftenreihe des Instituts für Photogrammetrie, Heft 14. Universität Stuttgart.

Toomey, M. A. G. 1988. The Alberta Digital Elevation Model. *In: IAPRS*, vol. 16. ISPRS, Kyoto. Presented paper, Kommission III, ISPRS-Kongress.

Part VI.

Appendix

A. Background Knowledge

A.1. Application System Architecture – Background and Terms

SCOP++ can be seen as program system of distributed architecture, and organized within the application frame "XX". In this architecture, modules belong to three major groups: XX itself, Agents, and Servers. XX consists of the user interface UX, the graphics interface GX, and SX to provide system and communication services. Servers are to perform algorithmic computation tasks in the background. Processing logics specific to the application is the main functionality of the Agent modules – in short: of the Agents.

cmL/cmF is part of UX. User actions, whether via the GUI or via cmL/cmF, are handled by UX, and the information thus acquired is passed from UX to the Agents via "SX messaging". Acting on these messages, Agents communicate with the user applying means as provided by UX – i.e. by sending, via SX, messages to UX. And also, Agents let Servers perform computations as needed.

SX messaging may be asynchronous or synchronous. Acting on asynchronous messages is delayed, meaning that the sender is not blocked by waiting for "readies" to asynchronous messages: the user interface remains active; the processes initiated are performed asynchronously, giving users the impression of parallel processing. The opposite is true for synchronous messages.

When controlled via the GUI, modules in XX-based application systems work asynchronously. Controlled via cmL/cmF, SX messaging and thus the entire processing become synchronous.

Error and informative messages are sent to the user from all three module groups: concerning certain aspects (syntax, ranges) of user actions, messages are sent immediately by UX; Agents are sending messages on encountering logical problems with user requests, and Servers may send messages on problems of processing data: indirectly, via the corresponding Agents. Agents may ask users to take decisions in ambiguous situations.

The architecture is laid out for "lazy processing": computations are only performed if needed for some product actually required by the user – e.g. for displaying some derived image. For this and for other purposes, Agents register processing states, and exchange (via SX) messages among themselves.

The user interface is dynamical: different elements such as buttons, entry fields and the similar may be created on the fly, according to Agent reactions to user actions. The corresponding information is sent by the Agents to UX via SX messages. UX builds up an unexpectedly large and complex database – the "uxTable" – to store these requests. The uxTable enables UX to handle user entries without displaying this information on the

screen via the GUI, meaning that UX is capable of batch in the sense of background processing. In the latter case, cmL/cmF provides the ways for user entries, rather than the GUI. Nevertheless, cmL/cmF is fully functional in interactive GUI modes of operation also, thus adding important capabilities to the user interface.

B. WINPUT

B.1. Preliminary

This is a description of a data acquisition scheme "WINPUT" for digitally recorded terrain height information, as recommended by the Institute of Photogrammetry of the Technical University of Vienna.

B.1.1. Single record

Each record contains a point number (=code) and coordinates x, y, z corresponding to Easting, Northing, Elevation.

The order of code, x, y, and z within the recording can be chosen at convenience. By default, this order is assumed to be: code, x, y, z.

Data format has to remain constant for a set of data (=model). The following characters are permissible:

Digits: `1 2 3 4 5 6 7 8 9 0`
Sign: `+ -`
Decimal point: `.`

Of these characters, integer (e.g. 123 or -3456) or real numbers (e.g. 123. or -3.456) may be composed.

Point numbers are, positive integer numbers (e.g.12345). If point numbers are recorded as real numbers (which is allowed), they will be truncated to their resp. integer values. Point numbers may contain 3 to 8 digits.

Coordinates may be recorded as integer or real numbers.

SCOP allows reals to contain up to 14 digits; coordinate differences, however, should not exceed values of max. 7 significant digits. (Otherwise accuracy may be lost depending on the computer applied). Example: (maximum x - minimum x) = 123456.789 yields an accuracy 0.05 only.

B.2. Data Organisation

It is a basic concept with WINPUT, to store all information concerning the recording itself together with the terrain point recordings. For this purpose, "control header recordings", and "terrain point recordings" are used. Certain groups of these recordings are delimited by "delimiter recordings".

Coordinates in delimiting records are disregarded.

Examples:

Beginning of a model:	99...91
Starting group with control points:	99...94
Starting terrain point recordings:	99...98
End of model:	99...99

The delimiter recording 99...91 and the model number has always to be at the beginning of a model.

The order of control header groups is arbitrary.

Terrain point recordings contain the data acquired in profiles, along contour lines, points distributed at random, break lines etc., as coded in the first two digits (left justified) of the point number. These codes are standards in WINPUT, e.g.

```
10000000    X   Y   Z     profiles in y-direction (WINPUT code 10)

51000020    X   Y   Z     closed break line No. 20 (WINPUT code 51)
```

Point numbers within terrain point recordings must have the same number of digits as within the control header recordings.

Terrain point recordings with a 0 point number are disregarded.

B.2.1. Data organization summary

```
99999991 ...... ...... ......       model beginning
  MODNUM ...... ...... ......
99999992 ...... ...... ......       units and scales
     MXY ...... ...... ......
     UXY ...... ...... ......
      UZ ...... ...... ......
99999993 ...... ...... ......       model extensions
    CODE   X      Y      Z
       .
       .
       .

    CODE   X      Y      Z
99999994 ...... ...... ......       control points (coordinates
                                     in the model system)
     CNR   X      Y      Z
       .
       .
       .

     CNR   X      Y      Z
99999995 ...... ....... ......      point density characteristics
                                     of terrain point recordings
  OFFSET ...... ...... ......
 DENSITY ...... ...... ......
99999998 ...... ...... ......       terrain point recordings
CC...LNR   X      Y      Z
```

```
        .
        .
        .

CC...LNR   X      Y      Z
99999999 ...... ...... ......       model end

99999991 ...... ...... ......       next model beginning
...
99999999 ...... ...... ......       model end
```

B.3. Control Recordings

B.3.1. (Next) model

```
99999991 ...... ...... ......  delimiting recording: model beginning
  MODNUM ...... ...... ......  model number
```

Recording this group of the control header record is compulsory in WINPUT: each model must start with it.

B.3.2. Model scales and units

```
99999992 ...... ...... ......
     MXY ...... ...... ......         model scale  \
                                                    for the XY-plane
     UXY ...... ...... ......         units        /
      MH ...... ...... ......         model scale  \
                                                    for the heights
      UH ...... ...... ......         units        /
```

Model scale and units can be recorded separately for the XY-plane and for elevations.

Scale: the point number field of the corresponding recording contains right justified the scale denominator (e.g. 10.000 for 1:10.000 or 1 for 1:1 in case of a non-integer denominator (e.g.1:6666.66...) it suffices to assure that the ratio (scale in XY):scale in H) remains undistorted. Example: MXY= 10 000, MH=6666.66...can be exactly processed with MXY=3, and MH=2).

Units: of the recorded coordinates, defined by one right justified digit in the point numer field:

0 meters
1 decimeters
2 centimeters
3 millimeters
4 tenths of millimeters (mm/10)
5 hundredths of millimeters (mm/100)

Note that the above numbers correspond to the number of decimals if coordinates are expressed in meters.

Example: Scale in the xy-plane 1: 20 000 Recording hundredths of a millimeter (mm/100) in xy-coordinates and decimeters (m/10) in heights:

```
99999992 ...... ...... ......
   20000 ...... ...... ......
       5 ...... ...... ......
       1 ...... ...... ......
       1 ...... ...... ......
```

If the recordings for heights (MH and UH) are missing in this control header group, the programs will take MXY resp. UXY as such.

Example: both xy-coordinates and heights recorded in the ground coordinate system, with the units being centimeters (m/100)

```
99999992 ...... ...... ......
       1 ...... ...... ......
       2 ...... ...... ......
```

The control header group 99...92 plays a crucial role in the following cases:

- scale in the xy-plane differs from the scale of the heights
- units of the xy-coordinates differ from those of heights
- no absolute orientation necessary but the units recorded differ from the units of the ground control system.

B.3.3. Model extension

```
99999993 ...... ...... ......
    CODE   X       Y       Z       \
       .                            \
       .                             \
                                        limiting points
       .                             /
                                    /
    CODE   X       Y       Z       /
```

The CODE in the point number field defines the logical position of the point within the model: (see table on next page). These points define the extension of the digitized area. It is not the extension of orthophoto nor of DTM.

1-digit CODE right justified	Explanation	Scheme
1	left lower corner point	2
2	left upper corner point	3
3	right side limiting or corner point	1
9	point of a limiting polygon	

Example:

```
                                                2 *------------
99999993 ...... ...... ......                     |           |
       1 100000 100000 100000                     |           * 3
       2 100000 125000 100000                     |           |
       3 115000 120000 100000                     |           |
                                                1 *------------
```

or

```
                                                  ------------* 3
99999993 ...... ...... ......                     |           |
       1 100000 100000 100000                     |           |
       3 115000 125000 100000                     |           |
                                                1 *------------
```

B.3.4. Model coordinates of control points

necessary if an absolute orientation is to be performed – now or in the future, for any, maybe unforseen, purpose.

```
99999994 ...... ...... ......
     CNR   X      Y      Z
       .
       .
       .

     CNR   X      Y      Z
```

The delimiting recording 99…94 marks the beginning of the group with the model coordinates of control points. There may be recorded a maximum of 50 control points per model. Control points must not have numbers greater than 999…0 (Collision with delimiting recordings).

B.3.5. Point density characteristics

```
99999995 ...... ...... ......
  OFFSET ...... ...... ......
 DENSITY ...... ...... ......
```

The specifications OFFSET and DENSITY in the point number field describe data density and have different meanings depending upon the sort of data acquisition techniques applied:

	main part of terrain point recordings belongs to:		
	profiles	contour lines	points distributed at random
OFFSET	distance between profiles	height step between contour lines	average point destance
DENSITY	(average) distance between the points of a profile	(average) distance between the points a long contour lines	may be omitted or will be handled as a dummy

OFFSET and DENSITY have to be expressed in the units specified past the delimiter recording 99...2. In this, scale in z and height units apply to height step definition for contour lines. Scale and units for the xy coordinates apply in all other cases.

Example: Profiles recorded in scale 1:20 000, units mm/100, profile offset (in model) 2 mm, point density within the profiles 1 mm:

```
99999992 ...... ...... ......
   20000 ...... ...... ......
       5 ...... ...... ......
         .
         .
         .

99999995 ...... ...... ......
     200 ...... ...... ......          = 200 mm/100
     100 ...... ...... ......          = 100 mm/100
```

Example: Contour lines recorded in a model with xy-scale 1:20 000, xy-units mm/100, heights recorded in the ground control system (1:1), units of heights m/10 (decimeters), height step between contours 10 m, point density a long the contours 2 mm:

```
99999992 ...... ...... ......
   20000 ...... ...... ......
       5 ...... ...... ......
       1 ...... ...... ......
```

```
         1 ...... ...... ......
         .
         .
         .

  99999995 ...... ...... ......
       100 ...... ...... ......          = 100 dm
       200 ...... ...... ......          = 200 mm/100
```

B.3.6. Terrain point recordings

```
99999998 ...... ...... ......
```

This delimiting recording must be written at least once.

Terrain point recordings contain in the field of point numbers a code describing the point recorded. Such coding enables the application of WINPUT data for the creation of digital height models of a fine precision (e.g. by SCOP).

The general form of a terrain point record is

```
CC...LNR    X       Y       Z
```

CC marks the first two (left justified) digits in the point number field: the code mentioned (see Table CC of codes). LINR stands for "line number" and denotes the right justified digits in the point number field.

NOTE: LNR is, in general, common to a series of terrain point records (line; points distributed at random should be grouped to such "lines", as well). Therefore, automatic point numbering (point numbers increased automatically by some recording devices) must not be used as LNR!

IMPORTANT:	in its current version, SCOP cannot correctly process structures containing more than some 10000 points. (e.g. a break line may not have more than 10000 points)

Thus the point number field of a terrain point record can be represented as

```
 _______________
| | | | | | | | |
|C|C|.|.|.|L|N|R|
|_|_|_|_|_|_|_|_|
  |           |________line number
  |___ code
```

In SCOP, LNR may contain a maximum of 4 right justified digits. Digits between CC and LNR are disregarded in SCOP.

B.3.7. Model end

```
99999999 ...... ...... ......
```

This delimiting recording marks the end of a model, and it must be recorded at least one.

B.3.8. Table CC of codes in terrain point records

Group	Type		CC
	point disregarded by the program		00
Profiles	profiles with x=const.		10
	profiles with y=const.		11
	DTM grid points		12
	Alignment		15
	Cross Section		16
Contour lines	in open terrain		20
	uncertain (e.g. in forest)		21
Points distributed at random	bulk data		30
Spot heights	high		31
	low		32
Formlines		open	40
		closed	41
Breaklines		open	50
		closed	51
Breaklines, which are also used as borderlines	omit right	open	52
		closed	53
	omit left	open	54
		closed	55
Borderlines	omit right with heights	open	60
		closed	61
	omit right without heights	open	62
		closed	63
	omit left with heights	open	64
		closed	65
	omit left without heights	open	66
		closed	67
	outer borderline without heights	closed	68
	exclusion line without heights	closed	69
Special points	off-terrain points		70
Elements of the situation			80 ... 89
Control codes	deletion code		90
	delimiter code		99

In coding border lines (60-67) and border lines coinciding with breaklines (50-55) care should be taken as to precisely define where the representation (e.g. of contour lines on a

map to be derived) should be omitted and vice versa. In this, *left* and *right* are determined in looking from the beginning of the line in the direction of its proceeding.

B.4. Example

```
99999991 000000 000000 000000   begin model
00004243 000000 000000 000000   model number
99999992 000000 000000 000000   header group scales and units
00010000 000000 000000 000000
00000005 000000 000000 000000
00000001 000000 000000 000000
00000002 000000 000000 000000
99999993 000000 000000 000000
00000001 100000 100000 113645
00000002 100000 112500 111847
00000003 109500 100255 120345
99999994 109500 100255 120345   header group control points
00004635 103885 112733 112536
00004673 103383 118376 111238
00004344 109388 112736 109980
00004372 102828 112233 102222
42435546 113567 123425 112727
99999995 113567 123424 112727
00000100 113567 123425 112727
00000200 113567 123425 112727
99999998 113567 123425 112727
       .
       .
       .
       .

99999998 113567 123425 112727   starting terrain point recordings
10000000 117546 112772 102992
10000000 117546 126489 102959
       .
       .

51000123 192462 043119 103011
       .
       .

30000000 121462 105726 134562
99999999 121462 105726 134562   end of model
```

The corresponding FORTRAN FORMAT specification:

```
FORMAT(D8.0,3(1X,D6.0))
```

C. List of Files created by SCOP++

In this section, a list of files is given which are created by SCOP++. When installing SCOP++, the user is requested to specify a directory which is used for project specific data. Further on, this directory will be called the scopProjects-directory. In this section, only files created in the scopProjects-directory are listed.

The following subdirectories will be created in the scopProjects-directory:

C.1. The Directory `topdb`

This directory is created by the system for storing tables of the topographic database using the following files and directories:

- `TDM.CMD`: temporarily used command file
- `TDM.LOG`: temporarily used logging file
- `TDM.RPT`: temporarily used report file
- `tmp.Fa?????`: These are temporary files which are usually deleted after program termination. However, if they cannot be removed after abnormal termination, the user can delete these files.

Files which must not be deleted:

- the subdirectories `cfg`, `cvt`, `dpm`, `grf`, `mkt`, `sys`, `vcb` including files of type `*.top`: In these files, the content of the database is stored. These files must not be deleted.

C.2. Project Directories

Each time a new project is created, a directory with the project's name is created. Per default, project directories are located as sub-directories of the scopProjects-directory. Nevertheless, the operator can specify any other location. The following files and directories are created in the project directory (suppose the name of the project is "samplePrj"):

- `samplePrj.spr`: The spr-file is the SCOP++ project file including setup data of the project.
- `samplePrj.log`: Logfile of project.
- `samplePrj.err`: Error file of project.

- `SAMPLEPR.TOP`: The TOP-file includes the table of the topographic database. The name of the table is created automatically by the system.
- `samplePrj.cmF`: The cmF-file is the command file of SCOP++. When creating a new project, a cmF-file including an empty AutoStart-procedure is created as template for manual editing.
- `samplePrj.fra`: Plotfile of frame.
- `winput`: temporarily used frame data.

C.2.1. Overlay Directories

Each model overlay or image overlay has its own directory. In these directories, many temporary files are created. Many of them are not deleted by the system in order to allow quick reloading of a project together with all its derived products. In the following, all files are listed in two categories: whether they should not be deleted or may be deleted.

Model Overlays

The overlay directory of a model overlay (e. g. with the overlay name "terrainModel") includes:

Files which should not be deleted:

- `terrainModel.stp`: The stp-file of an overlay includes all setup data. If it is not available, default settings will be chosen. So if it is deleted all your settings will be discarded!

Files which may be deleted: All subsequently listed files will be reproduced by SCOP++, so you may delete them although this will cost you some time, if you want to look at the products again.

- `terrainModel.dtm`: The dtm-file stores the digital terrain model. Many applications need that file.
- `terrainModel.sh0, terrainModel.sh1`: Shaded views are stored in these files.
- `terrainModel.zc0, terrainModel.zc1`: Z-coded views are stored in these files.
- `terrainModel_.all`: Terrain data used to generate the digital terrain model.
- `terrainModel_osh0.(b)wnp`: ObjectShapes within the terrain data.
- `terrainModel._L_`: Terrain data to be displayed within the limits window.
- `terrainModel_preThin.(b)wnp`: Output of Thinning in Data Preprocessing.
- `terrainModel_preFill.(b)wnp`: Output of Filling in Data Preprocessing.
- `terrainModel_preJoin.(b)wnp`: Output of Filling in Data Preprocessing merged with the input of Filling i.e. the original data or the output of Thinning, if used.

- `structures.(b)wnp`: Extracted non-bulkpoints of terrain data (robust filtering).
- `terrainModel_fil.(b)wnp`: Output of an eventual FillVoidAreas-step (robust filtering).
- `fsteps\stepi_elg.(b)wnp`: Output file of i'th EliminateBuildings step (robust filtering). Ground points (used for further calculation).
- `fsteps\stepi_elb.(b)wnp`: Output file of i'th EliminateBuildings step (robust filtering). Building points (sorted out).
- `fsteps\stepi_tho.(b)wnp`: Output file of i'th ThinOut step (robust filtering).
- `fsteps\stepi.dtm`: DTM file of i'th Filter or Interpolate step.
- `fsteps\stepi_grd.(b)wnp`: File of ground points of i'th Filter step.
- `fsteps\stepi_veg.(b)wnp`: File of vegetation points of i'th Filter step.
- `fsteps\stepi_sog.(b)wnp`: Output file of i'th SortOut step (robust filtering): Ground points (used for further calculation).
- `fsteps\stepi_sov.(b)wnp`: Output file of i'th SortOut step (robust filtering): Vegetation points (sorted out).
- `fsteps\stepi_edt.(b)wnp`: Output file of i'th Edit step (robust filtering).
- `fsteps\stepi_fil.(b)wnp`: Output file of i'th FillVoidAreas step (robust filtering).
- `terrainModel.dsm`: Slope model.
- `terrainModel_fcd?.wnp`: Feature code classes file (Quality).
- `terrainModel_ne.dtm`: Nearest model (Quality).
- `terrainModel_ne.zg?` and `terrainModel_ne.zc?`: Z-coding of nearest quality.
- `terrainModel_in.zg?` and `terrainModel_in.zc?`: Z-coding of internal quality
- `terrainModel.pis`: Temporarily used isoplot.
- `terrainModel.gln`: Temporarily used file for gridlines.
- `terrainModel.gpt`: Temporarily used file for gridpoints.

Scratch files which should be deleted by SCOP++:

- `terrainModel.out`: The out-file is the SCOP protocol file. The relevant protocols are listed in the log file of the project, if this option is used.
- `terrainModel.shz`: Temporary file for shaded views.
- `terrainModel_?.mix`: Mixed raster graphics files.
- `SHD.lut`: Temporarily used look-up table.
- `ZCO.lut`: Temporarily used look-up table.
- `stepi.edt.1`: Original file (before editing) of i'th Edit step (robust filtering).
- `lasctat.dnr`: Statistical info about last filter step (roubust filtering).

- terrainModel.cmd: temporarily used command file for server processes
- algnxy.dat and algnxyz.dat: Temporarily used alignment data.
- KA55 and KA40: Temporarily used alignment and cross section data.
- lprev.plt: Temporarily used plotfile for alignment.
- xprev.plt: Temporarily used plotfile for cross sections.

Image Overlays

The overlay directory of an image overlay (e. g. with the overlay name "orthoPhoto" and with an image data file op10.tif) includes:

Files which should not be deleted:

- orthoPhoto.stp: The stp-file of an overlay includes all setup data. If it is not available, default settings will be chosen.
- op10.pyr: an image pyramid description file.
- op10_??.*: higher level image data files of the pyramid (the extensions are for instance *.tif)
- op10.tfw: Geo-coding of tif-files
- op10.jgw: Geo-coding of jpg-files
- op10_??.tfw, op10_??.jgw: Geo-coding of pyramid levels
- op10.tp0, op10.tp1: Results of image processing operations
- op10.t0w, op10.t1w: Geo-coding of above files
- op10_??.tp0, op10_??.tp1: pyramid level files of image processing results
- op10_??.t0w, op10_??.t1w: Geo-coding of above files

Files which may be deleted:

- orthoPhoto._L_,
- orthoPhoto._L0,
- orthoPhoto._L1,
- orthoPhoto._0w,
- orthoPhoto._1w: temporary files for the limits window
- orthoPhoto.sts: temporary file for displaying image statistics
- op10.bup: back-up-pyramid descrption file used for "undo" image processing operation
- orthoPhoto*.zwi: temporary files for displaying transfer functions

Perspective Views

The perspective view directory of a project with name "myView" includes:

Files which should not be deleted:

- `myView.stp`: The stp-file of a perspective view includes all setup data. If it is not available, default settings will be chosen.

Files which may be deleted:

- `myView_grid.ppe`: intermediate file with view of gridlines,
- `myView_silh.ppe`: intermediate file with view of silhouette lines,
- `myView_names.ppe`: intermediate file with annotations,
- `myView_frame.ppe`: intermediate file with frame,
- `<overlayName>.pp0`: intermediate files with images from main graphics,
- `<overlayName>.pp1`: intermediate files with images from main graphics,
- `<overlayName>_*.pp0`: intermediate files with images from main graphics,
- `<overlayName>_*.pp1`: intermediate files with images from main graphics.

D. End User License Agreement

End User License Agreement

IMPORTANT, READ CAREFULLY. THIS END USER LICENSE AGREEMENT ("AGREEMENT") IS A LEGAL AGREEMENT BETWEEN YOU AND TRIMBLE NAVIGATION LIMITED OR ITS AFFILIATES ("Trimble") and applies to the Trimble Software product, including any accompanying printed materials and any "online" or electronic documentation (collectively, the "Software"). This Agreement will also apply to any Software error corrections, updates or upgrades, if any, that are subsequently furnished by Trimble, unless such are accompanied by different license terms and conditions which will govern their use. BY CLICKING "YES" IN THE ACCEPTANCE BOX , OR BY INSTALLING, COPYING OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE UNUSED SOFTWARE TO THE PLACE FROM WHICH YOU OBTAINED IT FOR A REFUND. NOTWITHSTANDING THE FOREGOING, IN THE EVENT THAT A WRITTEN, EXECUTED AGREEMENT HAS BEEN ENTERED INTO BY YOU (OR THE ENTITY YOU REPRESENT) AND TRIMBLE WITH RESPECT TO THE SOFTWARE, YOUR USE OF THE SOFTWARE SHALL BE GOVERNED BY SUCH WRITTEN AGREEMENT, AND NOT BY THIS LICENSE AGREEMENT.

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold.

1. License and Restrictions

Subject to the terms and conditions of this License Agreement, and timely payment by You of the applicable license fee, Trimble grants You the limited, non-exclusive, non-transferable right and license for the applicable license term, to use the Software in the Territory on the applicable number of servers by the applicable number of concurrent users, in object code format only. You may only use the associated documentation for your internal business use in accordance with this License Agreement. "Territory" means the country in which the Software was delivered to You, provided that if the Software was delivered to a country in the European Economic Area ("EEA"), it may be used in any country in the EEA. The applicable license fee, license term, number of servers and number of concurrent users shall be set forth in the Trimble invoice for the Software, provided that if this is an Academic License or an Evaluation License, the special terms and conditions for such licenses set forth below shall apply to your use of the Software.

Use of the Software is controlled by a license key provided to Licensee by Trimble ("License Key"), which specifies; (i) the identity of the Software, including version number, (ii) the number of concurrent users permitted, (iii) the "node" or specific server or other computer on which the Software may run ("Key Server"), (iv) whether the license is "node-locked" and accessible only from the Key Server, or "floating access" and available across a Local Area Network that includes the Key Server, (v) the codes, which Licensee must input to initialize use of the Key Server(s), and (vi) the expiration dates of Evaluation Licenses and annual licenses, or the regeneration date, if any, of the License Key for perpetual licenses. In the event that Trimble discontinues licensing the Software, Licensee will be provided with an unlimited License Key.

"Academic License" means a license, under which an academic institution obtains a copy of the Software solely for study, instruction or non-commercial research. Academic Licenses are made available to a university or professional academic institution recognized or accredited by the local Ministry/Department of Education or other accredited agency (hereinafter "Academic Institution"). Academic Institution may allow use of the Software only by its academic personnel and/or by

students, neither of which acquire any rights therein. Academic Institution agrees to regularly provide Trimble with a detailed summary of its usage of the Software. Academic Institution grants Trimble the exclusive license for the commercial use of Rule Sets developed using the Software. Academic Institution agrees not to publish any Rule Sets without Trimble's prior written consent.

"Evaluation License" means use of the Software for evaluation purposes only, in a non-production environment.
If You have obtained an Evaluation License, You may use the Software for non-commercial, internal evaluation purposes for a period of thirty (30) days, or such other period as may be set forth on the applicable Trimble invoice.

The Software is delivered in object code only. Licensee shall not reverse compile, disassemble or otherwise reverse engineer the Software, except where, and only to the extent that, such prohibition is not permitted under applicable law. To the extent any applicable mandatory laws give you the right to perform any of the aforementioned activities without Trimble's consent in order to gain certain information about the Software for purposes specified in the respective statutes (e.g., interoperability), you hereby agree that, before exercising any such rights, you shall first request such information from Trimble in writing detailing the purpose for which you need the information. Only if and after Trimble, at its sole discretion, partly or completely denies your request, may you exercise such statutory rights. Without Trimble's prior written consent, Licensee may not provide the Software to a third party on a temporary basis and/or use the Software for the benefit or purposes of a third party whether by means of lease, loan, data processing services (e.g. "fee for service"), time sharing arrangements or otherwise. In addition, Licensee will not run the Key Server on a virtual machine based platform (e.g. VMware).

2. Ownership, Confidential Information and Agreement not to Assert

All rights in and to the Software, including all intellectual property rights therein and thereto, belong to Trimble and its licensors, and Trimble and its licensors hold and retain title to each copy of the Software. Licensee shall not copy or modify the Software, except that Licensee may copy the Software for the sole purpose of backup as long as all copyright and other notices are reproduced and included on the backup copy.

Licensee acknowledges that the Software constitutes the valuable confidential information and trade secrets of Trimble. Accordingly, Licensee shall at all times, both during the term of this License Agreement and thereafter keep in trust and confidence all the Software, and shall not disclose the same to any third party without Trimble's prior written consent.

As a condition of the rights granted to Licensee under this Section 1, Licensee irrevocably and perpetually agrees not to assert against Trimble or any of its current or future direct or indirect licensees or sub-licensees, distributors and/or resellers (collectively, "Trimble Licensees") any patent, or part thereof, copyright, trade secret or any other intellectual property right embodied, in whole or in part, in any Rule Set written by or on behalf of Licensee (including any Rule Set written by Trimble for use by Licensee) ("Licensee Rule Set"), to the extent that any Rule Set developed by Trimble or any Trimble Licensee infringes or misappropriates, in whole or in part, any patent, copyright, trade secret or other intellectual property right embodied in a Licensee Rule Set. "Rule Set" means a computer software application based on Trimble's Cognition Network Language that is created using the Software development capabilities.

3. Term

This License Agreement is effective until the earlier of the expiration of any temporary License Key issued to Licensee or termination in accordance with this License Agreement. Licensee may terminate this License Agreement by ceasing use, and deleting all copies of the Software possessed by Licensee. Trimble may terminate this License Agreement if Licensee breaches any of the terms or conditions in this License Agreement, and this License Agreement shall in any event automatically terminate in the event of a breach by Licensee of any of its terms or conditions. Upon termination of this License Agreement for any reason, Licensee shall immediately cease use, and delete all of Licensee's copies, of the Software. All provisions of this License Agreement relating to disclaimers of warranties, limitation of liability, remedies, or damages, and Trimble's proprietary rights shall survive termination of this License Agreement.

4. Warranty and Warranty Disclaimers

Trimble warrants that for a period of thirty (30) days from the date of delivery of the software to Licensee, the Software will conform in all material respects in accordance with documentation provided with the Software. Licensee's exclusive remedy and Trimble's sole obligation in the event of a breach of the foregoing warranty shall be for Trimble, at its option, to correct or replace the non-conforming Software.

EXCEPT AS EXPRESSLY PROVIDED IN THIS SECTION 4, THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, AND TO THE GREATEST EXTENT PERMITTED BY APPLICABLE LAW, TRIMBLE EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES OF ANY KIND, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH LICENSEE. Trimble does not warrant that the functions contained in the Software will meet Licensee's requirements or that the operation of the Software will be uninterrupted or error-free.

5. Limitation of Liability

YOU ASSUME THE ENTIRE RISK AS TO RESULTS AND PERFORMANCE OF THE SOFTWARE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL TRIMBLE OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION OR DATA, CROP LOSS OR DAMAGE, OR ANY OTHER PECUNIARY LOSS), ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE, OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF TRIMBLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY EXCLUSIVE REMEDY PROVIDED IN THIS AGREEMENT.
IN NO EVENT SHALL TRIMBLE'S TOTAL LIABILITY IN CONNECTION WITH THIS AGREEMENT OR THE SOFTWARE, WHETHER BASED ON CONTRACT, WARRANTY,

TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EXCEED THE ACTUAL AMOUNT PAID TO TRIMBLE FOR USE OF THE SOFTWARE GIVING RISE TO THE CLAIM. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

6. Support

Licensee may purchase maintenance and support services for the Software by entering into an agreement with Trimble for the provision of such services.

7. Export Control and Restricted Rights Legend

Licensee may not directly or indirectly export or re-export, or knowingly permit the export or re-export of the Software (or portions thereof) to any country, or to any person or entity subject to United States export restrictions or any export and import control laws in the Territory in contravention of such laws and without first obtaining appropriate license.

Use, duplication, or disclosure of the Software by the United States Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

8. General

Except as otherwise stated herein, this License Agreement contains the entire agreement and understanding between the parties regarding the subject matter hereof, and replaces any prior written or oral understanding regarding such subject matter. Any different or additional terms or conditions contained in any Licensee purchase order are hereby rejected and shall not be deemed part of this License Agreement. Any attempt to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

9. Choice of Law

This License Agreement shall be governed by the laws of the State of California and applicable United States Federal law without reference to “conflict of laws” principles. The United Nations Convention on Contracts for the International Sale of Goods will not apply to this License Agreement. Exclusive jurisdiction and venue of any dispute or action arising from this License Agreement or the Software shall lie exclusively in the federal or state courts located in the County of Santa Clara, California. You hereby consent and agree not to contest such jurisdiction, venue and governing law.

Notwithstanding the above, if you acquired this product in Canada, this License Agreement is governed by the laws of the Province of Ontario, Canada. In such case each of the parties to this License Agreement irrevocably attorns to the jurisdiction of the courts of the Province of Ontario and further agrees to commence any litigation that may arise under this License Agreement in the courts located in the Judicial District of York, Province of Ontario. If you acquired this product in the European Union, this License Agreement is governed by the laws of The Netherlands, excluding

its rules governing conflicts of laws and excluding the United Nations Convention on the International Sale of Goods. In such case each of the parties to this Agreement irrevocably attorns to the jurisdiction of the courts of The Netherlands and further agrees to commence any litigation that may arise under this License Agreement in the courts of The Hague, The Netherlands. Trimble reserves all rights not expressly granted by this License Agreement.

10. Country Unique Terms

If You purchased a license to the Software in any Territory specified below (the "Local Territory"), this section sets forth specific provisions as well as exceptions to the above terms and conditions that apply in such Local Territory. To the extent any provision applicable to the Local Territory set forth below (the "Local Provision") is in conflict with any other term or condition in this agreement, the Local Provision will supersede such other term or condition with respect to any licenses purchased in the Local Territory.

Belgium and France

(a) Limitation of Liability (Section 5): The following replaces the terms of this section in its entirety:

Except as otherwise provided by mandatory law, Trimble's liability for any damages and losses that may arise as a result of the performance of its obligations in connection with this License Agreement is limited to the compensation of only those damages and losses proved and actually arising as an immediate and direct consequence of the non-fulfillment of such obligations (if Trimble is at fault), for a maximum amount equal to the charges You paid for the Software that has caused the damages. This limitation shall not apply to damages for bodily injuries (including death) and damages to real property and tangible personal property for which Trimble is legally liable.

UNDER NO CIRCUMSTANCES IS TRIMBLE, OR ANY OF ITS LICESORS AND/OR SOFTWARE DEVELOPERS, LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY: (i) LOSS OF, OR DAMAGE TO, DATA; (ii) INCIDENTAL OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; (iii) LOST PROFITS, EVEN IF THEY ARISE AS AN IMMEDIATE CONSEQUENCE OF THE EVENT THAT GENERATED THE DAMAGES; OR (iv) LOSS OF BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

The limitation and exclusion of liability herein agreed applies not only to the activities performed by Trimble, but also to the activities performed by its suppliers and Software developers, and represents the maximum amount for which Trimble as well as its suppliers and Software developers, are collectively responsible. This limitation shall not apply to damages for bodily injuries (including death) and damages to real property and tangible personal property for which Trimble is legally liable.

Germany and Austria

(a) Warranty (Section 4): The following replaces the terms of this section in its entirety:

Trimble warrants that the Software provides the functionalities set forth in the associated documentation ("Documented Functionalities") for one (1) year following delivery of the Software when used on the recommended hardware configuration. Non-substantial variation from the Documented Functionalities does not establish any warranty rights. THIS LIMITED WARRANTY DOES NOT APPLY TO SOFTWARE PROVIDED TO YOU FREE OF CHARGE (FOR EXAMPLE, UPDATES, OR ACADEMIC OR EVALUATION LICENSES) OR SOFTWARE THAT HAS BEEN ALTERED BY YOU, TO THE EXTENT SUCH ALTERATION CAUSED A DEFECT. To make a warranty claim, you must return, at Trimble's expense, the Software and proof of purchase to Trimble. If the functionalities of the Software vary substantially from the agreed upon functionalities, Trimble is entitled, by way of re-performance and at its own discretion, to repair or replace the Software. If that fails, you are entitled to a reduction of the license fee or to cancel this License Agreement.

(b) Limitation of Liability (Section 5): the following paragraph is added to this Section:

The limitations and exclusions specified in this Section will not apply to damages caused by Trimble intentionally or by gross negligence. In addition, Trimble shall be responsible up to the amount of the typically foreseeable damages from any damage which has been caused by Trimble or its agents due to the slightly negligent breach of a material contractual duty. This limitation of liability shall apply to all damage claims, irrespective of the legal basis thereof and in particular, to any pre-contractual or auxiliary contractual claims. This limitation of liability shall not, however, apply to any mandatory statutory liability under the product liability act, or to any damage which is caused due to the breach of an express warranty to the extent the express warranty was intended to protect You from the specific damage incurred. This clause shall not be intended to limit liability where the extent of liability is provided by mandatory law.

Italy

(a) Limitation of Liability (Section 5): the following replaces the terms of this section in its entirety:

Apart from damages arising out of gross negligence or willful misconduct for which Trimble may not limit its liability, Trimble' liability for direct and indirect damages related to the original or further defects of the Software, or related to the use or the nonuse of the Software or related to any case whatsoever for breach of the Agreement, shall be limited to the fees paid by you to Trimble for the Software or for the part of the Software upon which the damages were based.