Chapter 7

Operations

Note: This chapter is written as an enhancement to Chapter 7: Operations in the ILWIS 2.1 Reference Guide. It does *not* completely replace the mentioned chapter: instead you are invited to use both as one.

7.1 Visualization

7.1.1 Show map list as Color Composite

Select a map list in the Show Map List as Color Composite dialog box, and display three maps present in this map list together as an interactive color composite.

One map will be displayed in shades of red, one in shades of green and one in shades of blue. Putting three images together in one color composite can give a better visual impression of the reality on the ground, than by displaying one band at a time. Examples of color composites are false color (or IR) images and 'natural color' images.

By using an interactive color composite, you can easily change intervals, select other bands, etc. The resulting color composite is displayed in a map window, which can be saved as a map view. Interactive color composites are very suitable to be used as a background during sampling or during screen digitizing.

Your graphics board needs to be configured to use more than 256 colors, for instance High Color 16-bit, or True Color 24-bit (see Display Settings in Windows' Control Panel).

☞ When you want to create a permanent color composite, which can for instance be used as a raster drape over a 3D model, choose Color Composite from the Operations, Image Processing menu.

7.2 Raster Operations

7.2.1 Cross

Functionality

The Cross operation performs an overlay of two raster maps. Pixels on the same positions in both maps are compared; the occurring combinations of class names, identifiers or values of pixels in the first input map and those of pixels in the second input map are stored. These combinations give an output cross map and a cross table. The cross table includes the combinations of input values, classes or IDs, the number of pixels that occur for each combination and the area for each combination.

Input map requirements
- Both maps should have the same georeference.
- No restrictions on domain types.

Domain and georeference of output map and table
Cross creates an Identifier domain for the output map and table. This output domain obtains the same name as the output map and is filled with the combinations of class names, IDs or values of both input maps. When an input map has a class or ID domain in which the class names or IDs have codes, then these codes will appear in the output domain.
The output map uses the same georeference as the input maps.

Example

Input map 1:

B	B	C	D
A	A	C	D
A	A	C	C
B	C	D	A

Input map 2:

S	S	S	S
R	R	S	S
T	R	S	T
T	R	T	T

Output cross map:

BS	BS	CS	DS
AR	AR	CS	DS
AT	AR	CS	CT
BT	CR	DT	AT

In the picture of the output map above, read B×S instead of BS, etc.

Output cross table

Domain	Map1	Map2	NPix	Area
A * R	A	R	3	…
A * T	A	T	2	…
B * S	B	S	2	…
B * T	B	T	1	…
C * R	C	R	1	…
C * S	C	S	3	…
C * T	C	T	1	…
D * S	D	S	2	…
D * T	D	T	1	…

The cross table lists

Domain	the combination of class names, identifiers, values or group names of the first map with the second map's class names, IDs or values is returned as the output domain for the cross table. If class names or IDs in the input map have codes, then these codes will appear in the output domain.
Map1	the class name, identifier, or value of pixels in the first input map.
Map2	the class name, identifier, or value of pixels in the second input map.
NrPix	the number of pixels that occur as a combination.
Area	the areas of combinations as: NrPix * pixel size * pixel size.

Combinations with undefined values

When you use the Cross operation through the Cross dialog box, combinations with undefined values will by default not appear in the output cross table, i.e. undefined values are ignored. If an output cross table should also list combinations with undefined values, clear one or both of the *Ignore Undefs* check boxes in the dialog box. The combination Undefined value in the first input map and Undefined value in the second input map will never be listed.

Dialog box

Dialog box options

First map:	Select the first input raster map. Open the drop-down list box by clicking it and select a map, or directly drag a map from the Catalog into this box.
Ignore undefs:	Select this check box to ignore undefined values in the first input map: in the output cross table, combinations with undefined values in the first map will not be listed. Clear this check box when combinations with undefined values in the first map should be listed in the cross table.
Second map:	Select the second input raster map.
Ignore undefs:	Select this check box to ignore undefined values in the second input map; in the output cross table, combinations with undefined values in the second map will not be listed. Clear this check box when combinations with undefined values in the second map should be listed in the cross table.
Output table:	Type a name for the output cross table which will contain the combinations of the two input maps. This name will also be used for the output domain.
Show:	Select this check box if you want the output cross table to be directly displayed. Clear this check box if you do not want to see this table immediately: you simply define how the output table (and map) should be created.
Description:	Optionally, type a description for the output cross table (and map).
Output map:	Select this check box if you want to obtain a cross map containing the combinations of the two input maps. Subsequently, type a name for the output raster map. Clear this check box if an output cross map is not desired.

A dependent table is created; optionally, a dependent map can be created as well. Furthermore, an ID domain is created (same name as cross table) which contains the combinations of domain items of both input maps.

Command line

Cross can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTTABLE= `TableCross`(FirstInputMap, SecondInputMap)
OUTTABLE= `TableCross`(FirstInputMap, SecondInputMap, OUTMAP)
OUTTABLE= `TableCross`(FirstInputMap, SecondInputMap [, OUTMAP] , `IgnoreUndefs`)
OUTTABLE= `TableCross`(FirstInputMap, SecondInputMap [, OUTMAP] , `IgnoreUndef1`)
OUTTABLE= `TableCross`(FirstInputMap, SecondInputMap [, OUTMAP] , `IgnoreUndef2`)

where:

OUTTABLE	is the name of your output cross table.
OUTMAP	is an optional parameter to create an output cross map.
`TableCross`	is the command to start cross and produce an output cross table (optionally with an output cross map).
FirstInputMap	is the name of your first input raster map (any domain).
SecondInputMap	is the name of your second input raster map (any domain).
`IgnoreUndefs`	an optional parameter to ignore undefined values in both maps; in the output cross table combinations with undefined values will not appear. When the parameter is not used, the output cross table will also show combinations with undefined values.
`IgnoreUndef1`	an optional parameter to ignore undefined values of the first input map.
`IgnoreUndef2`	an optional parameter to ignore undefined values of the second input map.

Alias:

OUTMAP= `MapCross`(FirstInputMapName, SecondInputMapName, OUTTABLE)

OUTMAP	is the name of your output cross map.
`MapCross`	is the command to start cross and produce an output cross map and an output cross table.

For other parameters, see above.

When the definition symbol `=` is used, a dependent output table and/or a dependent map are created; when the assignment symbol `:=` is used, dependency links are immediately broken after the output table and/or map have been calculated.

7.2.2 Glue raster maps

Functionality

The Glue raster maps operation glues or merges two or more georeferenced input raster maps into one output raster map. The output map then comprises the total area of all input maps. The domains of the input maps are merged when needed. With the Glue raster maps operation, you can thus merge two or more adjacent or partly overlapping raster maps (i.e. make a mosaic) or glue smaller raster maps onto a larger one.

When the input raster maps have attribute tables, also the tables will be automatically merged; for more information see the Glue tables operation.

Input maps: In the dialog box, you can select 2, 3, or 4 input raster maps. On the command line, you can specify as many input maps as you like. The input maps may be purely adjacent to one another, partly overlapping, or totally overlapping. When the input maps are (partly) overlapping, the input maps form a pile of maps on top of each other.

Map on top: When the pile of input maps are (partly) overlapping the same area, you have to decide which map should be considered as the map on top. When for a pixel a value is found in the map on top, that value will appear in the output map. When the undefined value is found in the map on top, the operation will look in the map 'below' it. Undefined pixels thus act as being transparent and provide 'openings' to enable the operation to find a value in the map below the current one.

In the dialog box, you can select the check box 'Last Map on Top' to order the input maps as:
- the map selected last on top,
- the map selected one but last below that one,
- until the map selected first.

On the command line, you can use the `Replace` option to this end.

Then, for each pixel, the operation will do:
- when a value is encountered in the map you selected last: that value will appear in the output map;
- when the undefined value is encountered in the map you selected last: the operation will look in the map that was selected one but last;
  - when a value is encountered in the map you selected one but last: that value will appear in the output map;
  - when the undefined value is encountered in the map you selected one but last: the operation will in the map that was selected second but last, etc.

Only when no value is found at all, the output pixel will be assigned the undefined value. Note: when the output map uses the Image domain, this means value 0; when the output map uses a `Picture` domain or the `Color` domain, this means color (0,0,0), i.e. black.

First input map
The georeference of your first input map will be used to construct a new georeference for the output map. The output georeference will always be sized in such a way that all input georeferences will fit in it. However, the georeference of your first input map will directly determine the pixel size and the coordinate system for the output georeference. The output georeference will obtain the same name as the output raster map; the output georeference is a georeference of type submap.

Domain combinations
You can always merge maps that use the same domain, and maps that use domains of the same type. In some cases, you can also merge maps of different domain types. The list below shows the possible combinations of input domain types and also shows the output domain type.

In	In	Out
Image	Image	Image
Image	Value	Value
Image	Color	Color
Value	Value	Value
Value	Color	Color
Class	Image	Color
Class	Value	Color
Class	Class	Class
Class	Color	Color
ID	ID	ID
Picture	Picture	Picture
Picture	Color	Color
Color	Color	Color
Bool	Bool	Bool
Bit	Bool	Bool
Bit	Bit	Bit

For more information, see also the section Domain of output map below.

Georeferences
When all input raster maps use the same georeference, the output raster map will also use that georeference. When the input georeferences are different, a new georeference (type submap) will be automatically created. The output georeference is based on the georeference of your first input map but the georef size will be extended so that all input georeferences fit in the output georeference. The georeference of your first input determines furthermore the pixel size and coordinate system of the new georeference. If necessary, all input maps will be automatically resampled to the output georeference; the resampling is done according to the Nearest Neighbour method. See also the section Georeference of output map below.

Example
To show a certain `landuse` type (of a class map) on a satellite image:

- Select an image that you want to use as background and stretch the image with the Stretch operation.
- To obtain for instance a `landuse` map showing only agricultural units, perform MapCalc statements like:
  ```
  Agri= iff ((landuse="agriculture") or
  (landuse="agriculture (irrigated)"), landuse, "?")
  ```
  Explanation: if map landuse is classified as agriculture, then have these classes remain, else assign undefined.
- Use the Glue Raster Maps operation to merge the map Agri with your stretched image. The results may look like below.
- The result of the Glue Raster Map operation is shown in Figure 1 below.

Figure 1: Agricultural land use over a satellite image. The dark gray part is agriculture with irrigation; the light gray part is rainfed agriculture.

For presentation and printing purposes, you could of course also use the Mask Polygons operation, use as mask `agri*` . Then, display the output polygon map on top of the satellite image in a map window.

Input map requirements
The domains of the input maps should be mergeable, see the list of domain combinations above.
All input raster maps must have a georeference, which is not georeference `None`.

Domain of output map
The domain type for the output map can be found in the list of domain combinations above. Here some extra information will be given on how the system deals with the domains for the output maps.

- The output map will use *system domain Image* when you are combining images with each other.
- The output map will use *system domain Value* when you are combining value maps with value maps or images with value maps. The smallest minimum and the largest maximum of all input maps determine the value range for the output map; the precision is the smallest precision of the input maps.
- The output map will use a *class domain* when all input maps use a class domain. When all input maps use the same class domain, then the output map will also use this class domain. When the input maps use different class domains and when the class names in the input domains are different, then all class names of all input domains will be merged automatically into a new output domain. You can choose whether the new domain should be stored as a separate object, or whether it should be stored by the output map (internal domain). Also the representations of the input class domains will be merged into a new output representation; the new representation is either stored as a separate object or by the output map (internal representation).
- The output map will use an *ID domain* when all input maps use an ID domain. The same procedure is followed as for class maps. There are no representations involved.
- The output map will use a *Picture domain* when all input maps use a Picture domain. When all input maps use the same Picture domain, the output map will also use this Picture domain. When the input maps use different Picture domains and when the colors in the input Picture domains are different, then the colors of all input domains will be merged automatically into a new output Picture domain. You have to keep in mind though that a Picture domain can only contain a maximum of 256 colors. Thus, when the total number of all input colors is 256 or less, the colors of the input pictures will be retained in the output picture. The new Picture domain will always be stored by the output map (internal domain) and not as a separate object.
- The output map will use *system domain Color* when you are combining images, value maps, class maps or pictures with a map that uses the Color domain, and when you are combining class maps with images or class maps with value maps. For class maps and pictures, the output colors will be retrieved from the representation of the input domains; for images and value maps, the output color will be retrieved from system representation Gray is used.

Georeference of output map
When all input maps use the same georeference, that georeference will also be used for the output map. When the input maps use different georeferences, then a new georeference (type submap) will be automatically created for the output map. The new georeference will always use the pixel size and coordinate system of your first input map; the size of new the georeference will be such that all input maps fit in it. The new georeference will obtain the same name as the output map. If necessary, all

input maps are resampled with the Nearest Neighbour method to this output georeference.

Usually, the input georeferences will use the same coordinate system, which covers the whole area already. In case input georeferences use different but compatible coordinate systems (e.g. different projections), the coordinate system of the first input map is used for the output georef, and a coordinate transformation is performed for the other input maps.

☞ When input maps are resampled to a new output georeference, this will be done with the Nearest Neighbour method. When working with value maps with different georeferences, and when you would like to calculate with the output value map of the Glue Raster maps operation, you will obtain better interpolation results from the Glue Raster maps operation, when:
- you first make one new large georeference,
- resample your maps to this new georeference by using the Resample operation with the Bilinear or BiCubic interpolation method, and
- only when all maps use the same new large georeference, use the Glue Raster Maps operation.

☞ When you want to combine class maps with ID maps, you first have to convert either the class domain into an ID domain or the ID domain into a class domain. In fact, it is quite easy to convert Class and ID domain(s) to each other: open the Domain Properties dialog box of the domain that you want to convert, and click the Convert to Classes or the Convert to IDs button. When converting IDs to classes, you can create a representation class for your map; when converting from classes to IDs, you will loose your representation class.

☞ When you want to combine a class map with one image, it is advised to first stretch the image; this will improve the contrast in the output map with the `Color` domain.

☞ When you want to combine a class map with multiple images, it is advised to first use the glue raster maps operation only with all images, then stretch the output image and finally use the glue raster maps operation again to merge the class map in it. This will improve the contrast in the output map with the `Color` domain.

Notes
- When merging class maps or ID maps, by default an internal domain is created for the output map to reduce the number of separately stored domains. Internal domains are stored internally in the output map. If you like, you can select the New Domain check box in the dialog box and specify a new name for the output domain if you want the output domain to be stored as a separate object.
- The maximum number of columns for any output map is 64000 for a 1-byte output map, 32000 for a 2-byte output map, 16000 for 4-byte output map and 8000 for an 8-byte output map.

For more information on internal domains and representations, refer to How to open internal domains/representations.

Dialog box

Dialog box options

Number of input maps:	Select the number of raster maps that you want to glue or merge together (2, 3, or 4). In the dialog box, the number of input raster maps is limited to four. On the command line, this limitation is not present.
First input map:	Select the first input raster map. Open the list box and select the desired input map, or drag a raster map directly from the Catalog into this box.
Second input map:	Select the second input raster map.
Third input map:	Optionally, select a third input raster map.
Fourth input map:	Optionally, select a fourth input raster map. You can always glue raster maps of the same domain type.
Last map on top:	Select this check box when for overlapping pixels the values, class names, IDs, or colors of the last map should be used. Then, for overlapping pixels, the values of the last map will appear on top of the other maps; undefined values in the last map will act as openings for the map under it, etc. Clear this check box when for overlapping pixels, the pixels of the first map should be used.
New domain:	Only when merging Class or ID maps that do not use the same domain: select this box if you want to store the output domain as a separate object (recommended). Subsequently, type a name for the new domain. Clear this check box to obtain an output Class or ID domain, which is stored by the output map (internal domain).
Output raster map:	Type a name for the output raster map that will contain all input maps.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created. Furthermore, a new georeference is automatically created for the output map (same name as output map), and, when needed and specified, a new domain is also created.

Command line

The Glue raster maps operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapGlue`(FirstInputMapName, SecondInputMapName [, MoreInput Maps])
OUTMAP = `MapGlue`(FirstInputMapName, SecondInputMapName [, MoreInput Maps] , `Replace`)
OUTMAP = `MapGlue`(FirstInputMapName, SecondInputMapName [, MoreInput Maps] , NewDomain)
OUTMAP = `MapGlue`(FirstInputMapName, SecondInputMapName [, MoreInput Maps] , NewDomain, `Replace`)

where:

OUTMAP	the name of your output raster map.
`MapGlue`	the command to start the Glue raster maps operation.
FirstInputMapName	the name of the first input raster map.
SecondInputMapName	the name of the second input raster map.
MoreInputMaps	optionally, you can specify more input raster map names, delimited by commas.
`Replace`	an optional parameter to use for overlapping pixels the values, class names, IDs or colors of the last input map. When this parameter is not used, then the values, class names, IDs or colors of the first input map will be used for overlapping pixels.
NewDomain	an optional parameter in case of merging Class or ID maps that do not have the same domain, to specify a name for the new output domain in which all input domain items will be merged. When input Class or ID domains are not the same and this parameter is not specified, the new output domain in which all input domain items are merged will be stored by the output map (internal domain).

On the command line, you can specify as many input raster maps as you like; i.e. you can merge as many raster maps as you like. When using the dialog box of this operation, only two, three or four input maps can be merged at a time.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated. When Class or ID maps are merged that do not have the same domain, and the NewDomain parameter is used, a new output domain is also created. Furthermore, a georeference submap is created with the same name as the output map.

Algorithm

First, the operation checks whether the georeferences and the domain types of the input maps are compatible. If so, the operation can be performed. For more information on mergeable domain types, see Glue raster maps: functionality.

Output georeference

A new georeference submap is created with the same name as the output map.

- The pixel size of the first input map is used and the georeference is extended in X and Y direction so that all input maps fit into the output georeference.
- If needed, all input maps are resampled to this output georeference. The nearest neighbour resampling method is used in the same way as when resampling the values of a single raster map to a new georeference (see Resample).
- Usually, the input georeferences will use the same coordinate system, which covers the whole area already. In case the input georeferences use a different but compatible coordinate system (e.g. different projections), the coordinate system of the first input map is used and a coordinate transformation is performed for the other maps.

Output domain

- When merging maps with the same domain, the output map will also use that domain.
- When merging value maps, the value range for the output map is determined by the smallest minimum and the largest maximum of all input maps; the precision is the smallest precision of the input maps.
- When merging a value map with an Image, the output map always uses system domain `Value`. The smallest minimum and the largest maximum of the input maps determine the value range for the output map; the precision is the best precision of the input maps.
- When merging Class maps or when merging ID maps that do not have the same domain, a new output domain is created which contains all domain items, i.e. all classes or all IDs. The new domain is either stored as a separate object, or is stored by the output map (internal domain).
- When merging Pictures, a new output domain and a new representation are created which are both stored by the output map (internal domain and internal representation); the representation contains all colors of the input maps with a maximum of 256.
- When merging an image or a class, ID, or value map with a `Color` map, the output map will use the Color domain
- When merging a class map with an image or when merging a class map with a value map, the output map will use the `Color` domain.

Overlapping pixels

- By default, the last map appears on top of the first map.
- Undefined pixels in the map on top are considered transparent and act as openings.

7.3 Image Processing

7.3.1 Filter

Command line

The Filter operation can be directly executed by typing the following expression on the command line of the Main Window.

OUTMAP= `MapFilter`(InputMapName, *FilterName* | *FilterExpression*)

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
FilterName	`avg3x3` | `binmajor` | `conn8to4` | `d2fdx2` | `d2fdxdy` | `d2fdy2` | `dfddn` | `dfdup` | `dfdx` | `dfdy` | `dilate4` | `dilate8` | `edgesenh` | `inbnd4` | `inbnd8` | `laplace` | `lifegame` | `majority` | `majundef` | `majzero` | `med3x3` | `med5x5` | `outbnd4` | `outbnd8` | `peppsalt` | `shadow` | `shrink4` | `shrink8` | *name of user-defined linear filter*
FilterExpression	`FilterLinear`(*rows,cols,expression*) | `Average`(*rows,cols*) | `RankOrder`(*rows,cols,rank[,threshold]*) | `Median`(*rows,cols[,threshold]*) | `Majority`(*rows,cols*) | `ZeroMajority`(*rows,cols*) | `UndefMajority`(*rows,cols*) | `Pattern`(*threshold*) | `FilterStandardDev`(*rows,cols*)

☞ For more information and some examples of using standard and user-defined filters on the command line, refer to Filters : user-defined filters.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated.

7.3.2 Filter types

Based on the algorithm used by filters, the following types of filters can be distinguished:

- Linear filters
  - Gradient or derivative filters
- Rank order and median filters
- Majority filters
- Binary filters

- Pattern filters
- Standard deviation filters

7.3.2.1 Linear filters

Linear (convolution) filters consist of a matrix with values and a gain factor. When considering a linear filter of size 3x3, the 9 matrix values are multiplied with 9 pixel values in the input raster map, this is summed and then multiplied with the gain factor. The result is assigned to the center pixel in the output map.

Standard linear filters
The following standard filters are linear filters,: AVG3X3, EDGESENH, LAPLACE, and SHADOW. Also the standard gradient filters are linear filters: DFDX, DFDY, DFDDN, DFDUP, D2FDX2, D2FDY2, D2FDXDY.

User-defined linear filters
You can create, edit and store your own linear filters, for instance from the Filter: dialog box, through the File menu in the Main window or by clicking the NewFilter item in the Operation-list. For more information, refer to the Create Filter and Edit Filter dialog boxes. You need to specify the size of your filter, fill out the values in the matrix and specify a gain factor. Experienced users may wish to experiment with the definition of a linear filter by an expression on the command line. For more information on using standard and user-defined linear filters, refer to User-defined linear filters.

Furthermore, you can interactively define an Average filter in the Filter dialog box or by an expression on the command line. For more information, refer to User-defined average filters.

7.3.2.2 Gradient or derivative filters (technical information)

Linear (convolution) filters consist of a matrix with coefficients and a gain factor. When considering a linear filter of size 3×3, the 9 matrix coefficients are multiplied with 9 pixel values in the input raster map, this is summed and then multiplied with the gain factor. The result is assigned to the center pixel in the output map.

The following standard filters are known as gradient filters or derivative filters: DFDX, DFDY, DFDN, DFDUP, D2FDX2, D2FDY2, D2FDXDY. For each group of pixel values considered, they calculate the first or second derivative in one or more directions. Derivative filters are often used in relation to slope calculations.

The standard derivative filters mentioned above have a typical size of 1×5, 5×1, or 5×5. Of course, you can also create your own linear gradient filters.

This topic is intended for people who would like to understand the mathematical background of the coefficients in the matrices of derivative filters. First, 3×3 filters will be described, then 5×5 filters.

3×3 filters

Introduction

The simplest mathematical situation is represented when using a 3×3 filter, however these filters may not be exact enough to calculate slopes. A 3×3 filter uses the 9 input values to calculate a value for the center pixel in the output map.

To do calculations with a 3×3 filter, a local coordinate system (X,Y) is defined around the current center pixel, as:

(-1, 1)	(0, 1)	(1, 1)
(-1, 0)	(0, 0)	(1, 0)
(-1, -1)	(0, -1)	(1, -1)

Both X and Y can have the values -1, 0, and 1.

When calculating the first derivative only in the x-direction, y remains 0, and x can have value -1, 0, or 1.

To calculate derivatives, a continuous function is needed. The input pixel values are to be described by a function `f(x)` where:

f_0 = input pixel value of the center pixel; x=0
f_{-1} = input pixel value of the pixel to the left of the center pixel; x=-1
f_1 = input pixel value of the pixel to the right of the center pixel; x=1

Formulas

As continuous function, a polynomial function is used. With 9 known values, a second order polynomial can be fitted through these points.

$$f_{xy} = a_{00} + a_{10}\,x + a_{20}\,x^2 + a_{01}\,y + a_{11}\,xy + a_{21}\,x^2y + a_{02}\,y^2 + a_{12}\,xy^2 + a_{22}\,x^2y^2 \quad (1)$$

When we are only interested in the first derivative in x-direction, formula 1 can be simplified by substituting y with 0 to:

$$f_x = a_0 + a_1\,x + a_2\,x^2 \quad (2)$$

The function for the first derivative equals:

$$df/dx = f'x = a_1 + 2a_2\,x \quad (3)$$

The second derivative equals:

$$d^2f/dx^2 = f''x = 2a_2 \quad (4)$$

Because we are interested in the derivatives at the central pixel where x=0, in formulas 3 and 4, x can be substituted with 0:

$$df/dx = f'_0 = a_1 \quad (5)$$
$$d^2f/dx^2 = f''_0 = 2a_2 \quad (6)$$

To find f_{-1}, f_0 and f_1, in formula 2, x is substituted with values -1, 0, and 1:

$$f_{-1} = a_0 - a_1 + a_2 \quad (7)$$
$$f_0 = a_0 \quad (8)$$
$$f_1 = a_0 + a_1 + a_2 \quad (9)$$

Then, by elimination, a_1 and a_2 are found:

$$a_1 = (f_1 - f_{-1}) / 2 \quad (10)$$
$$a_2 = (f_{-1} - 2f_0 + f_1) / 2 \quad (11)$$

A 3×3 first derivative filter for the x-direction will thus read:

0	0	0
-1	0	1
0	0	0

Gain factor = 1/2 = 0.5

A 3×3 second derivative filter for the x-direction will thus read:

0	0	0
1	-2	1
0	0	0

Gain factor = 1/2 = 0.5

To calculate the second derivative in both the x-direction and the y-direction, f"=d²f/dxdy, formula 1 is needed. After the substitution of all 9 coordinates in the equation, and solving them, the results are:

$$d^2f / dxdy = a_{11} \quad (12)$$
$$a_{11} = (f_{1,1} + f_{-1,-1}) - (f_{-1,1} + f_{1,-1}) / 4 \quad (13)$$

A 3×3 second derivative filter for both the x-direction and the y-direction will thus read:

-1	0	1
0	0	0
1	0	-1

Gain factor = 1/4 = 0.25

5×5 filters

Calculation of matrix coefficients for 5×5 filters follows the same method as for 3×3 filters. Although 5×5 filters are a little bit more complicated, they will produce more accurate results.

Again a local coordinate system is used around the current center pixel as:

(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,0)	(-1,0)	(0,0)	(1,0)	(2,0)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)

Both x and y can have the values -2, -1, 0, 1, and 2.

The polynomial function f_x and its derivatives are:

$$fx = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x \quad (14)$$
$$df/dx = f'x = a_1 + 2a_2 x + 3a_3 x^2 + 4a_4 x^3 \quad (15)$$
$$d^2f/dx^2 = f''x = 2a_2 + 6a_3 x + 12a_4 x^2 \quad (16)$$

By substituting x in formulas 15 and 16 with 0, the previous equations 5 and 6 are obtained again:

$$df/dx = f'_0 = a_1 \quad (5)$$
$$d^2f/dx^2 = f''_0 = 2a_2 \quad (6)$$

Substituting x in formula 14 with values -2, -1, 0, 1, and 2 gives f_{-2}, f_{-1}, f_0 , f_1 and f_2:

$$f_{-2} = a_0 - 2a_1 + 4a_2 - 8a_3 + 16a_4 \quad (17)$$
$$f_{-1} = a_0 - a_1 + a_2 - a_3 + a_4 \quad (18)$$
$$f_0 = a_0 \quad (19)$$
$$f_1 = a_0 + a_1 + a_2 + a_3 + a_4 \quad (20)$$
$$f_2 = a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4 \quad (21)$$

With some restructuring, the following equations are obtained:

$$f_1 + f_{-1} = 2a_0 + 2a_2 + 2a_4 \quad (22)$$
$$f_1 - f_{-1} = 2a_1 + 2a_3 \quad (23)$$
$$f_2 + f_{-2} = 2a_0 + 8a_2 + 32a_4 \quad (24)$$
$$f_2 - f_{-2} = 4a_1 + 16a_3 \quad (24)$$

The matrix coefficients for a 1×5 filter, which calculates the first derivative in x-direction, are found by the elimination of a_3: subtracting equation 23 eight times from equation 25. This results in:

$$a_1 = (f_{-2} - 8f_{-1} + 8f_1 - f_2) / 12 \quad (26)$$

A 1×5 first derivative filter for the x-direction will thus read:

1 -8 0 8 -1
Gain factor = 1/12 = 0.0833333

This is exactly what ILWIS uses for the DFDX filter.

The matrix coefficients for a 1×5 filter, which calculates the second derivative in x-direction, are found by the elimination of a_4: subtracting equation 22 sixteen times from equation 24 and substituting f_0 for a_0. This results in:

$$a_2 = (-f_{-2} + 16f_{-1} - 30f_0 + 16f_1 - f_2) / 24 \quad (27)$$

A 1×5 second derivative filter for the x-direction will thus read:

-1 16 -30 16 -1
Gain factor = 1/24 = 0.0416667

This is exactly the D2FDX2 filter.

7.3.2.3 Rank order and median filters

Rank order filters have a certain size, but do not have any matrix values or gain factor. A rank order filter of size 3x3 for example, examines 9 pixel values of the input map at a time, sorts the values from small to large, and selects for the output value that value which is encountered at a certain rank order number. So one value of the pixel values examined becomes the output value, without any calculation performed on the values itself.
When a threshold is set, the value of the center pixel will only be replaced with the new value if the difference between the original and new value is smaller than or equal to the threshold.

Standard rank order filters
Standard rank order filters are the median filters: MED3x3, MED5x5. For each 9 pixels considered, the MED3 filter always assigns the value of rank 5 to the center pixel in the output map. For each 25 pixels considered, the MED5 filter always assigns the value of rank 13 to the center pixel in the output map. The median filters can for instance be used to smooth an image.

User-defined rank order and median filters
In the Filter dialog box as well as on the command line, you can define your own rank order filter and median filter. For a rank order filter, specify the size of the filter, the rank order number and an optional threshold. In this way you can obtain for example the minimum or maximum value of a number of pixels. For median filters, specify the size of the filter and an optional threshold. For more information and some examples, refer to User-defined rank order and median filters.

7.3.2.4 Majority filters

For each group of pixels considered in the input map, a majority filter assign the predominant (=mostly frequently occurring) value or class name of these to the center pixel in the output map. Undef-majority filters only do this when the center pixel in the input map is undefined; zero-majority filters only do this when the value of the center pixel in the input map is 0.

Undef-Majority filters are often used as a post-classification operation to reduce the number of undefined pixels in the output map of an image classification.

Standard majority filters
For each 3×3 pixels considered, the standard MAJORITY filter assigns the predominant value or class name to the central pixel in the output map. If no predominant value is found, for instance when all 9 input pixels have a different value or class name, the value or class name encountered first is used as output.

The standard undef-majority filter (MAJUNDEF) will only assign the predominant value or class name to the central pixel in the output map if the central pixel in the input map is undefined.

The standard zero-majority filter (MAJZERO) will only assign the predominant value or class name to the central pixel in the output map if the central pixel in the input map has value zero.

User-defined majority filters
In the Filter dialog box as well as on the command line, you can define your own majority filters: specify the size of the filter and whether a condition has to be used. For more information, refer to user-defined majority filters.

7.3.2.5 Binary filters

Binary filters regard the input map as a binary map. This means that zero values are regarded as zero, and all other values as one. Depending on the central pixel value and its 8 neighbours, the filter produces a zero or one as output values. Binary filters are widely used for morphologic filtering.

The 9 binary pixels examined by a binary filters are put in a special order or bit position as below (where 0 means last position, 1 the one but last position, etc.):

5	6	7
4	8	0
3	2	1

This results in a number of 9 binary digits (when only lower right pixel is true: 000000010).

Thus depending on the position of true pixels a unique number is obtained. To decide whether the central pixel should be assigned a 0 or 1, this number is looked up in a table, which is present in each binary filter itself.

Standard binary filters
The standard binary filters are: BINMAJOR, CONN8TO4, DILATE4, DILATE8, SHRINK4, SHRINK8, INBND4, INBND8, OUTBND4, OUTBND8, PEPPSALT, LIFEGAME.

User-defined binary filters
There are no possibilities in ILWIS to define your own binary filters. However, you could copy an existing binary filter (*.FIL), edit it with an ASCII editor like Notepad and thus create your own binary filter.
Only Pattern filters, which work more or less the same as binary filters, can be user-defined via the Filter dialog box or the command line.

7.3.2.6 Pattern filters

With a pattern filter you can detect: **areas** where pixels have more or less the same value, **locations** where the values of all neighbours are largely different from the center pixel (outliers), and the **directions** in which differences between a center pixel and its neighbours are found. A pattern filter always works in a 3×3 environment and works on images and other raster maps with a value domain.

Whether or not any of the 8 neighbours is considered to have more or less the same value as the center pixel, is determined by the threshold value that you have to specify. When the absolute difference between a neighbour and the center pixel is smaller than or equal to the threshold value, the answer is true. For each true neighbour, a certain bit is set. The value assigned to the center pixel in the output map is the bit-wise combination of all true neighbours. For more information, refer to Example pattern filters.

The pattern filter works according to the following rules:
- All neighbours are false: the final output value is 0, which means that there are large differences between the central pixel and all its neighbours (outlier).
- All neigbours are true: the output value is 255, which means that there are small differences between the central pixel and its neighbours (area),
- All other output values bit-wise represent the directions in which differences are found.

Standard pattern filters
There is no standard pattern filter stored on disk.

User-defined pattern filters
A pattern filter always works in a 3×3 environment. In the Filter dialog box as well as on the command line, you can define a pattern filter; specify the threshold value, which has to be used. For more information and some examples, refer to User-defined pattern filters.

7.3.2.7 Standard deviation filters

For each group of pixels in the input map, a standard deviation filter calculates the standard deviation and assigns this value to the center pixel in the output map.

Standard deviation filters can be useful for radar images. The interpretation of radar images is often difficult: you can not rely on spectral values because of backscatter (return of the pulse sent by the radar). This often causes a lot of 'noise'. By using a standard deviation filter, you may be able to recognize some patterns.

The formula to calculate the standard deviation reads:

$$\text{stddev} = \sqrt{\frac{\sum (x_i - \bar{x})^2}{r \cdot c - 1}}$$

Where:

x_i	are the individual pixel values of the input map considered by the filter is the mean of the pixel values considered by the filter
r	is the size of the filter in rows
c	is the size of the filter in columns

Standard standard deviation filters
There is no standard standard deviation filter stored on disk.

User-defined standard deviation filters
In the Filter dialog box as well as on the command line, you can define a standard deviation filter: specify the size of the filter. For more information and some examples, refer to User-defined standard deviation filters.

7.3.2.8 User-defined filters

Besides using the ILWIS standard filters, you can:
- create, edit and store your own linear filters,
- define a filter in the Filter dialog box by specifying some parameters,
- define a filter on the command line by an expression.

Creating linear filters
You can create, edit and store your own linear filters, for example for example by clicking the create button in the Filter dialog box, by selecting Create Filter command from the File menu of the Main window, or by double-clicking the NewFilter item in the Operation-list.
You can specify the size of the linear filter, insert your own values in the matrix and specify a gain factor. For more information, refer to the Create Filter dialog box and the Edit Filter dialog box.

Defining a filter in the Filter dialog box
In the Filter dialog box, you can select any standard (predefined) filter.
Furthermore, you can define the following filters according to your wishes: Average filter, Rank Order filter, Median filter, Majority filter, Pattern filter, and Standard Deviation filter. You can specify the filter size, a rank or the threshold.

Defining a filter by an expression on the command line
Finally, advanced users can define an Average filter, a Rank Order filter, a Median filter, a Majority filter, a Pattern filter, a Standard Deviation filter or even a Linear filter by typing an expression on the command line of the Main window.

Restrictions of user defined filters
These restrictions apply to all user defined filters, except the pattern filter.

- Both the number of rows and the number of columns defining the filter size must be odd values.
- The product of *rows*×*cols* defining the filter size may not exceed 8000.

Restrictions of user defined filters (specific)
- When using the rank order filter, the minimum rank is 1 and the maximum rank cannot exceed the total size of the filter (*rows*×*cols*).

7.3.2.9 User-defined linear filters

Create, edit and store a linear filter on disk
A user-defined linear filter can be created:
- by clicking the create button in the Filter dialog box, or
- by choosing the Create Filter command from the Main window File menu, or
- by double-clicking the NewFilter command in the Operation-list.

The Create Filter dialog box and the Edit Filter window will appear. You can specify the size of the filter, fill out values in the matrix, and specify a gain factor.

Edit an existing user-defined linear filter
An existing user-defined linear filter can be edited:
- by clicking your filter with the right mouse button in the Catalog, and subsequently selecting the Open command from the context-sensitive menu,
- by choosing the Edit Object command from the Edit menu in the Main window, and subsequently selecting your filter.

The Edit Filter window will appear.

The standard linear filters cannot be edited.

User-defined average filters
Additional possibilities are provided to for user-defined average filters, via the Filter dialog box or via the command line, see Filters : user-defined average filters.

Selecting an existing linear filter in the Filter dialog box
In the Filter dialog box, you can select any standard linear filter or any linear filter you created yourself.
- Open the Filter dialog box,
- for Filter Type, select: Linear,
- for Filter Name, select: AVG3X3, D2FDX2, D2FDXDY, D2FDY2, DFDDN, DFDUP, DFDX, DFDY, EDGESENH, LAPLACE, Shadow, or your own linear filter.

Using existing linear filters on the command line
To use an existing linear filter from the command line, type the following expression on the command line of the Main window.

OUTMAP = `MapFilter`(InputMapName, *FilterName*)

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
FilterName	is either the name of a standard filter on disk, then fill out one of the following standard linear filter names: `av3x3` | `d2fdx2` | `d2fdxdy` | `d2fdy2` | `dfddn` | `dfdup` | `dfdx` | `dfdy` | `edgesenh` | `laplace` | `shadow` or the name of a linear filter which you created yourself.

Defining a linear filter on the command line (advanced)
Advanced users may wish to experiment with the definition of a linear filter by an expression. The syntax for the command line is:

OUTMAP= `MapFilter`(InputMapName, `FilterLinear`(*rows*, *cols*, expression))

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`FilterLinear`	is the command to define a linear filter.
rows	are the number of rows of your linear filter.
cols	are the number of columns of your linear filter.
expression	is an expression in which you can use x, y, and r to calculate the linear filter's matrix values, where:
x	distance to the center cell of the matrix in x-direction; the center cell of the matrix has position (0,0); the distance in x-direction increases to the right.
y	distance to the center cell of the matrix in y-direction; the center cell of the matrix has position (0,0); the distance in y-direction increases downwards.
r	Euclidean distance to the center cell of the matrix: $\sqrt{(x^2+y^2)}$

Examples of linear filter expressions are for instance:

```
FilterLinear(5,5,2x+y)
```

Defines a 5 by 5 shadow filter where the illumination is from the north-west.

```
FilterLinear(5,5,-2x-y)
```

Defines a 5 by 5 shadow filter where the illumination is from the south-east.

```
FilterLinear(5,5,iff(r>3,0,4-r))
```

Defines an average filter in which the matrix values decrease linearly, starting from the center cell of the matrix towards the borders of the matrix, according to Euclidean distance.

7.3.2.10 User-defined average filters

The standard average filter is AVG3X3; it calculates the average value of 9 pixel values considered. In the Filter dialog box as well as on the command line, you may define other average filters of any user-defined size.

Using the Filter dialog box

1. To use the standard Average filter AVG3X3:
   - Open the Filter dialog box,
   - for Filter Type, select: Linear,
   - for Filter Name, select: AVG3X3.

2. To define your own average filter:
   - Open the Filter dialog box,
   - for Filter Type, select: Average,
   - specify the size of the filter.

See also the example below.

Using the command line (advanced)

1. To use the standard Average filter AVG3X3, type on the command line of the Main window:

   OUTMAP = `MapFilter`(InputMapName,AVG3X3)

   where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Avg3x3`	is the name of the standard average filter.

2. To define your own Average filter, type the following expression on the command line of the Main window:

   OUTMAP = `MapFilter`(InputMapName,`Average`(*rows* , *cols*))

   where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Average`	is the command to define an average filter.
rows	are the number of rows of your average filter.
cols	are the number of columns of your average filter.

Example

To use an average filter which considers each **5 vertically neighbouring pixels**:

- in the Filter : dialog box, choose for Filter type: Average, specify 5 for the number of rows and 1 for the number of columns,
- on the command line, use a filter definition of: `Average(5,1)`.

7.3.2.11 User-defined rank order and median filters

Standard rank order filters are MED3x3 and MED5x5; these filters sort 9 respectively 25 pixel values and assign the 5th respectively the 13th value (median value) to the center pixel in the output map. In the Filter dialog box as well as on the command line, you may define other rank order and median filters of any user-defined size and assigning any user-defined rank order value to the central pixel.

Using the Filter dialog box

1. To use one of the standard rank order filters:
   - Open the Filter dialog box,
   - for Filter Type, select: Rank Order,
   - select the Predefined check box,
   - for Filter Name, select: Med3x3 or Med5x5.

2. To define your own rank order filter:
   - Open the Filter dialog box,
   - for Filter Type, select: Rank Order,
   - clear the Predefined check box,
   - specify the size of the filter,
   - specify the rank order value which should be assigned to the central pixel,
   - optionally specify a threshold: if the absolute difference between the result and the original value is larger than the threshold, the original value is kept; if the difference is smaller than or equal to the threshold, the ranked value is assigned.

   This ensures that larger differences between neighbouring pixels remain in the output, while small variations are removed.

3. To define your own median filter:
   - Open the Filter dialog box,
   - for Filter Type, select: Median,
   - specify the size of the filter,
   - optionally specify a threshold.

See also the example below.

Using the command line (advanced)

1. To use a standard Rank order filter, type one of the following expressions on the command line of the Main window:

OUTMAP = `MapFilter`(InputMapName, `Med3x3`)
OUTMAP = `MapFilter`(InputMapName, `Med5x5`)

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Med3x3`	is the name of the standard Median 3x3 filter.
`Med5x5`	is the name of the standard Median 5x5 filter.

2. To define your own Rank order filter, type one of the following expressions on the command line of the Main window:

OUTMAP = `MapFilter`(InputMapName, `Rankorder`(*rows,cols,rank*))
OUTMAP = `MapFilter`(InputMapName, `Rankorder`(*rows,cols,rank,threshold*))

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Rankorder`	is the command to define a rank order filter.
rows	are the number of rows of your rank order filter.
cols	are the number of columns of your rank order filter.
rank	is the rank order number that determines the output value.
threshold	when a threshold is set, the value of the center pixel will only be replaced by the new value if the difference between the original and new pixel value is smaller than or equal to the threshold.

3. To define your own Median filter, type one of the following expressions on the command line of the Main window:

OUTMAP = `MapFilter`(InputMapName,`Median`(*rows,cols*))
OUTMAP = `MapFilter`(InputMapName,`Median`(*rows,cols,threshold*))

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Median`	is the command to define a median filter.
rows	are the number of rows of your median filter.
cols	are the number of columns of your median filter.
threshold	when a threshold is set, the value of the center pixel will only be replaced by the new value if the difference between the original and new pixel value is smaller than or equal to the threshold.

Examples

1. To obtain the minimum of each 3×3 neighbouring pixels:
   - in the Filter dialog box, select for the Filter type Rank Order,
     clear the Predefined check box,
     specify 3 for the number of rows and 3 for the number of columns,
     specify 1 for the rank,
     and optionally specify a threshold.
   - on the command line, use a filter definition of: `Rankorder(3,3,1)`

Rank 1 means the smallest value of each 9 (3 by 3) pixel values examined will be used for the output pixel.

2. To obtain the maximum of each 7 horizontally neighbouring pixels:
   - in the Filter dialog box,
     select for the Filter type Rank Order
     clear the Predefined check box,
     specify 1 for the number of rows and 7 for the number of columns,
     specify 7 for the rank,
     and optionally specify a threshold.
   - on the command line, use a filter definition of: `Rankorder(1,7,7)`

   Rank 7 means that the largest value of each 7 (1 by 7) pixel values examined is used as output value.

3. To obtain the median value of each 5 horizontally neighbouring pixels:
   - in the Filter dialog box,
     select for the Filter type Median,
     clear the Predefined check box,
     specify 1 for the number of rows and 5 for the number of columns,
     and optionally specify a threshold.
   - on the command line, use a filter definition of: `Median(1,5)`.

☞ With Neighbourhood operators similar operations can be performed as with rank order filters: for example, the minimum, maximum, sum and predominant pixel value of each 9 pixels considered can be assigned as output value to the central pixel. For more information, refer to MapCalc special: neighbourhood operations.

☞ With the Aggregate Map operation, similar results can be obtained. Aggregate Map however moves from block to block in the map, while the Filter operation moves pixel by pixel.

7.3.2.12 User-defined majority filters

The standard majority filters are Majority, MajUndef, and MajZero; they work in a 3 by 3 environment and assign the predominant value of the surrounding pixels to the central pixel. In the Filter dialog box as well as on the command line, you may define other majority filters of any user-defined size.

Using the Filter dialog box

1. To use one of the standard Majority filters:
   - Open the Filter dialog box,
   - for Filter Type, select: Majority,
   - select the Predefined check box,
   - for Filter Name, select: Majority, MajUndef or MajZero.

2. To define your own Majority filter:
   - Open the Filter dialog box,
   - for Filter Type, select: Majority,
   - clear the Predefined check box,

- specify the size of the filter,
- in case you want to replace only central values which are undefined, select the Undefined check box.

See also the example below.

Using the command line (advanced)

1. To use a standard Majority filter, type one of the following expressions on the command line of the Main window:

   OUTMAP = `MapFilter`(InputMapName, `Majority`)
   OUTMAP = `MapFilter`(InputMapName, `MajUndef`)
   OUTMAP = `MapFilter`(InputMapName, `MajZero`)

   where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Majority`	is the name of the standard Majority filter.
`MajUndef`	is the name of the standard Undef-Majority filter.
`MajZero`	is the name of the standard Zero-Majority filter.

2. To define your own Majority filter, type one of the following expressions on the command line of the Main window:

   OUTMAP = `MapFilter`(InputMapName, `Majority`(*rows,cols*))
   OUTMAP = `MapFilter`(InputMapName, `UndefMajority`(*rows,cols*))
   OUTMAP = `MapFilter`(InputMapName, `ZeroMajority`(*rows,cols*))

   where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Majority`	is the command to define a Majority filter.
`UndefMajority`	is the command to define an Undef-Majority filter.
`ZeroMajority`	is the command to define a Zero-Majority filter.
rows	are the number of rows of your majority filter.
cols	are the number of columns of your majority filter.

Example

To use a majority filter which considers each **5 horizontally neighbouring pixels**:

- in the Filter dialog box, choose for Filter type: Majority,
  clear the Predefined check box, and
  specify 1 for the number of rows and 5 for the number of columns,
- on the command line, use a filter definition of: `Majority(1,5)`

7.3.2.13 User-defined pattern filters

With a pattern filter, you can detect areas where pixels have more or less the same value, outliers where the values of all neighbouring pixels are very different from the center pixel, and the directions in which differences between neighbouring pixel values are found. Whether or not 3×3 neighbouring pixels are considered to have more or less the same values, is determined by the threshold value that you have to specify.

There is no standard pattern filter stored on disk. A pattern filter always works in a 3×3 environment. In the Filter dialog box as well as on the command line, you can define the threshold value, which has to be used by the pattern filter.

Using the Filter dialog box
To use a pattern filter:

- Open the Filter dialog box,
- for Filter Type, select: Pattern,
- specify a threshold value which determines whether the absolute differences between the central pixel and its neighbours are considered large (edge) or small (area).

Using the command line (advanced)
To define a pattern filter, type the following expression on the command line of the Main window:

OUTMAP = `MapFilter`(InputMapName, `Pattern`(*threshold*))

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`Pattern`	is the command to define a pattern filter.
threshold	specify a value for the threshold which determines whether the absolute differences between the central pixel and its neighbours are considered large (outlier or direction of edge) or small (area).

7.3.2.14 User-defined standard deviation filters

There is no standard standard deviation filter stored on disk. You always have to define a standard deviation filter yourself, either through the Filter dialog box or by an expression on the command line of the Main window.

Using the Filter dialog box
To define your own standard deviation filter:

- Open the Filter dialog box,
- for Filter Type, select: Standard Deviation,
- specify the size of the filter.

Using the command line (advanced)
To define your own Standard Deviation filter, type the following expression on the command line of the Main window:

OUTMAP = `MapFilter`(InputMapName, `FilterStandardDev`(*rows,cols*))

where:

OUTMAP	is the name of your output map.
`MapFilter`	is the command to start the Filter operation.
InputMapName	is the name of your input map.
`FilterStandardDev`	is the command to define a standard deviation filter.
rows	are the number of rows of your standard deviation filter.
cols	are the number of columns of your standard deviation filter.

Example
To use a standard deviation filter which considers each **7 vertically neighbouring pixels**:
- in the Filter: dialog box, choose for Filter type: Standard Deviation, specify 7 for the number of rows and 1 for the number of columns,
- on the command line, use a filter definition of: `FilterStandardDev(7,1)`

7.3.3 Color composite

Functionality

A color composite is created by combining 3 raster images (bands/maps). One band is displayed in shades of red, one in shades of green and one in shades of blue.

A color composite can be created:
- to serve as a background image during sampling and subsequent image classification,
- for visual interpretation purposes, a printed color composite may be useful as a field map, but you can also use a color composite as a background image during on-screen digitizing, or
- for illustration purposes, for instance by using a color composite as a drape over a 3D model.

☞ This operation creates a permanent color composite map. When your graphics board is configured to use more than 256 colors, you can also interactively display a color composite by selecting the Show MapList as Color Composite command from the Operations, Visualization menu. (To customize the color depth to for instance High Color 16-bit or True Color 24-bit, use Display Settings in Windows' Control Panel). By creating an interactive color composite (very suitable for sampling or on-screen digitizing), you can easily change intervals, select other bands, etc. Another advantage is that besides maps with the Image domain, also maps with a Value domain are accepted. The resulting color composite is displayed in a map window. To store such a color composite, you can save the map window as a map view.

General information on color composites

- A color composite gives a visual impression of 3 raster bands. Putting the three bands together in one color composite map can give a better visual impression of the reality on the ground, than by displaying one band at a time. Examples of color composites are false color (or IR) images and 'natural color' images.
- The input pixel values of each band are measures of the amount of reflection in a certain wavelength interval. The values in the output color composite map just refer to certain colors; the output values themselves have no meaning.
- Before creating a color composite, you might filter the bands in order to increase sharpness of features of interest.

The Color composite operation offers several ways to create color composites:

- **Standard**:
  - Linear stretching: input values are linearly stretched; user-defined input intervals;
  - Histogram equalization: input values are equally divided over output colors; user-defined input intervals;
- **Dynamic** (Heckbert): input values are automatically distributed over a user-defined number of colors.
- **24 Bit RGB**:
  - Linear stretching: input values are linearly stretched; user-defined input intervals;
  - Histogram equalization: input values are equally divided over output colors; user-defined input intervals;
- **24 Bit HSI**: input values are interpreted as hue, saturation and intensity.

The different methods of creating a color composite are merely a matter of scaling the input values over the output colors. See below Color Composite: algorithm for the exact methods.

Input map requirements

The three input maps should use the Image domain. A georeference is not required for the input maps. If the maps do have a georeference, all input maps should use the same georeference.

Domain of output map

For a **standard** color composite: the operation always uses system Picture domain `ColorCmp` for the output color composite. This domain always uses system representation `ColorCmp`.
For a **dynamic** color composite: the operation creates a new domain (type Picture) for the output color composite and a new representation for this domain. This output domain and representation are always stored within the output map (internal domain and internal representation).
For a **24-bit** colors composite: the operation always uses the Color domain for the output color composite.

Georeference of output map

The output color composite always uses the same georeference as the input maps.

- If the user is interested in the image as a whole, it is best to use the Dynamic option. This usually results in a composite with good contrast. The Dynamic option does not take into account the structure of the input bands. Therefore, if the user is interested in specific intervals of the input bands, it is better to use the Standard option, using linear stretching. However, if the pixels that are of less interest can be masked, the user can calculate a Dynamic composite using only the pixels that are of interest.
- When there are more bands available than can be used to create a color composite (e.g. 7 TM-bands), you can first calculate the Optimum Index Factor (OIF); this may help you to decide which bands to select for a color composite.
- The reverse process of creating a color composite is color separation.
- For more information on internal domains and representations, refer to How to open internal domains/representations.

Dialog box

Dialog box options for 24 Bit Color Composite

24 bit:	Select this check box if you want to create a map that can be displayed in 24-bit graphic mode (domain Color). Clear this check box if you want to create a color composite, which uses a Picture domain.
RGB:	Creates a 24-bit color composite with Red, Green and Blue bands as input. See Standard Color Composite options below.
HSI:	Creates a 24-bit color composite with Hue, Saturation and Intensity bands as input.

You should only select these 24-bit options, if your graphic board is configured to use more than 256 colors, for instance High Color 16-bit or True Color 24-bit (see Display Settings in Windows' Control Panel).

Dialog box options for Standard Color Composite

Standard:	Select the Standard option button when you want to use the standard color composite representation, which has six shades of red, six shades of green and six shades of blue (total 216 colors).
Linear Stretching:	Select Linear Stretching if you want to obtain intervals of equal length (in terms of input values) for the output colors. Histogram equalization: Select Histogram Equalization if you want to obtain an equal number of pixels for the different output colors.
Percentage:	Select this check box to define input intervals by a percentage of pixels to be ignored on both sides of the input map's histogram. Clear this check box to specify input intervals by a minimum and maximum value of each input map.

Dialog box options for Dynamic (Heckbert) Color Composite

Dynamic:	Select the Dynamic option button when you want an automatic division of input values over a number of output colors.
Colors:	Enter a value for the number of colors of which the dynamic color composite should consist (integer value between 2 and 255).

General dialog box options

Red Band:	Select a raster map to be displayed in shades of red. Open the list box and select the appropriate raster map, or directly drag a raster map from the Catalog into this box.
Green Band:	Select a raster map to be displayed in shades of green.
Blue Band:	Select a raster map to be displayed in shades of blue.
Output raster map:	Type a map name for the output color composite.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created. When the Dynamic option is used, the output map will use an internal Picture domain, which has an internal representation.

☞ This operation creates a permanent color composite map. When your graphics board is configured to use more than 256 colors, you can also interactively display a color composite by selecting the Show MapList as Color Composite command from the Operations, Visualization menu. (To customize the color depth to for instance High Color 16-bit or True Color 24-bit, use Display Settings in Windows' Control Panel). By creating an interactive color composite, you can easily change intervals, select other bands, etc. The resulting color composite is displayed in a map window. To store such a color composite, you can save the map window as a map view.

Command line

Typing one of the following expressions on the command line of the Main Window can directly create a color composite:

OUTMAP= `MapColorComp`(MapList, *range1, range2*, *range3*)
OUTMAP= `MapColorCompLinear`(MapList, *range1, range2, range3*)
OUTMAP= `MapColorCompHistEq`(MapList, *range1, range2, range3*)
OUTMAP= `MapHeckbert`(MapList, *NrColors*)
OUTMAP= `MapColorComp24`(MapList)
OUTMAP= `MapColorComp24`(MapList, *range1, range2, range3*)

OUTMAP= `MapColorComp24Linear`(MapList)
OUTMAP= `MapColorComp24Linear`(MapList*, range1, range2, range3*)
OUTMAP= `MapColorComp24HistEq`(*MapList,range1, range2, range3*)
OUTMAP= `MapColorComp24HSI`(MapList)

Where:

OUTMAP	is the name of your output color composite.
`MapColorComp`	is the command start the Color composite operation using linear stretching.
`MapColorCompLinear`	is the command start the Color composite operation using linear stretching.
`MapColorCompHistEq`	is the command to start the Color composite operation using histogram equalization.
`MapHeckbert`	is the command to start the Color composite operation using the Heckbert dynamic algorithm.
`MapColorComp24`	is the command to start the 24-bit Color composite operation using linear stretching.
`MapColorComp24Linear`	is the command to start the 24-bit Color composite operation using linear stretching.
`MapColorComp24HistEq`	is the command to start the 24-bit Color composite operation using histogram equalization.
`MapColorComp24HSI`	is the command to start the 24-bit Color composite operation using hue, saturation, and intensity.
MapList	is the name of an existing map list which contains 3 raster maps with the `Image` domain, or the definition of a map list as: `mlist`(*ImageRed, ImageGreen, ImageBlue*). For more information, see the examples below.
range1..3	for each band, the intervals of input values to be used as *min:max* or as *perc*
min:max	minimum and maximum value determining range of input values, e.g. `20:200`; integer values between 0 and 255.
perc	percentage of input values to be ignored on both sides of the maps' histogram during linear stretch or histogram equalization; 0 ≤ real value < 50. When no ranges are specified for the MapColorComp24 or MapColorComp24Linear commands then all input values are used, i.e. no stretching. The MapHeckbert and the MapColorComp24HSI commands always use all input values.
NrColors	For Dynamic (Heckbert) color composites: the number of colors present in the output map (2-255).

All maps in the input map list must have the same georeference.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated.

Examples
A complete expression to calculate a color composite of bands tm4, tmb3, and tmb2, using linear stretching and ignoring 1% of all input values, might thus read:

```
OUTMAP = MapColorComp(mlist(tmb4,tmb3,tmb2),1,1,1)
```

A complete expression to calculate a dynamic color composite of bands tm4, tmb3, and tmb2, using 200 colors, might thus read:

```
OUTMAP = MapHeckbert(mlist(tmb4,tmb3,tmb2),200)
```

Algorithm

Standard color composites

Linear stretching of interval
The specified interval range per band is linearly divided into 6 classes of equal length with numbers 0 to 5. Since this is done for three bands, the number of possible combinations is 6×6×6 = 216. This is the number of different colors that will appear in the color composite.

Histogram equalization
The specified interval range (lower and upper boundary) per band is divided into 6 classes numbered 0 to 5, each of which has an equal area under the histogram. The number of different output colors is 6×6×6 = 216, just as it is for linear stretching.

Output colors
Each output color obtains an internal number of the system picture domain `ColorCmp`; the value is calculated as:

```
output = 36×red + 6×green + blue.
```

This implies that the output map will contain internal values between 0 and 215 (since 215 = 5 × 36 + 5 × 6 + 5). The values of this `ColorCmp` domain are always linked to system representation `ColorCmp`.

Dynamic color composites
A dynamic color composite is calculated using the Heckbert Quantization Algorithm. The Heckbert algorithm produces a color composite on the basis of the amount of variation in pixel values in the three input maps.

This algorithm first builds a three dimensional histogram, indicating how 'popular' any given value is in the images. All values fall in one box or cube. This histogram is then subdivided into smaller boxes or cubes: a division is made in the middle of the axis, which has the largest variation. This process continues until as many boxes are created, as there are output colors (number defined by the user, maximum 255). This algorithm attempts to create boxes, which have approximately equal popularity in the image. Then, colors are assigned to represent each box.

1. First, the input map values at 1% and 99% are determined. See also the Histogram operation.
2. The band with the largest variation in pixel values is selected, the total number of pixels for this band is calculated, and the band is divided into 2 halves, each containing half of the total number of pixels. The division leaves the other 2 bands intact. The result after one division is 2 so-called 'boxes': one band divided over the 2 boxes, and the other 2 bands complete in both boxes.
3. The program then searches for the next (part of a) band with the largest variation in pixel values. The total number of pixels on that (part of the) band is calculated, the (part of a) band is divided into 2 halves, so that each new part of the band contains half of the total number of pixels, and the other 2 bands are left as they were. The result after 2 divisions is 3 'boxes': each with (parts of) the red, the green, and the blue band.
4. The division process is repeated until the total number of 'boxes' reaches the number of user-defined colors desired for the output map.
5. Then colors are assigned to all boxes. For each box, weighted averages are calculated of the parts of the red band, green band and blue band covered by that box; the outcome values are the Red, Green and Blue values for that box. The calculation is repeated for all boxes.

An example of the first and second division is given in Figures 1, 2 and 3 below.

Example Heckbert algorithm
Three 1% histograms are calculated before the first division (see Figure 1).

Figure 1 shows that the pixel values range:
- from 31 to 98 in the first input map
- from 22 to 100 in the second input map and
- from 21 to 66 in the third input map.

Thus, the largest variation is found in the second input map. The total number of pixels on this band is divided in 2 so-called 'boxes': the first division is at pixel value 65 of band 2.

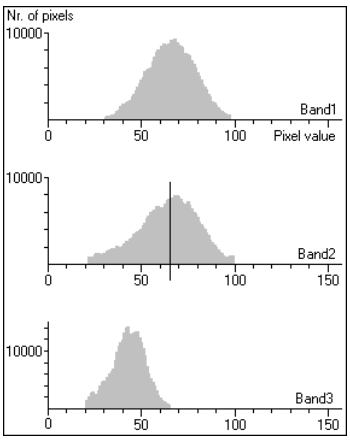


Figure 1: 1% histograms for Heckbert Color Composite

The result of the first division is represented in Figures 2 and 3. Figure 2 shows the histograms of one box; Figure 3 the histograms of the other. Enlarge the Help window by dragging its borders when you cannot see both figures next to each other.

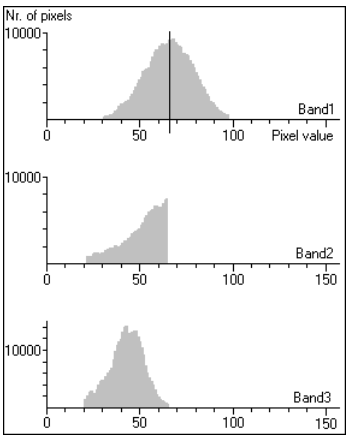


Figure 2:

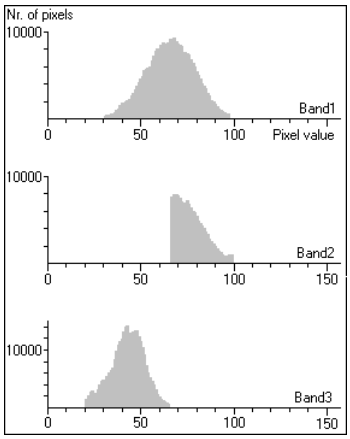


Figure 3:

Figure 2 shows that the pixel values range in box 1:

- from 31 to 98 in the first input map
- from 22 to 65 in the second input map, and
- from 21 to 66 in the third input map

Figure 3 shows that the pixel values range in box 2:

- from 31 to 98 in the first input map
- from 66 to 100 in the second input map, and
- from 21 to 66 in the third input map.

The next largest variation is found in the first input map in box 1. The total number of pixels on this band divided into 2 parts: the next division takes place at pixel value 66.

Output domain for dynamic composites
The operation creates a new domain (type Picture) for the output color composite and a new representation for this domain. This output domain and representation are always stored within the output map (internal domain and internal representation).

24-bit RGB color composites

Linear stretching of interval
Each input band is stretched to values between 0 and 255 using linear stretching using the user-defined intervals of the histograms.

Histogram equalization
Each input band is stretched to values between 0 and 255 using histogram equalization using the user-defined intervals of the histograms.

Output colors
The results are combined in a map with 4 bytes per pixel (4 bytes is 32 bits, thus 8 bits are not used). Each pixel in this map contains the red, green, and blue intensities of values between 0 and 255. It means that the possible number of output colors is $256 \times 256 \times 256 = \pm 16$ million.

24-bit HSI color composites

In this case no stretching is performed. In the output map, for each pixel the hue, saturation and intensity is converted to red, green, and blue intensities. The following relations exist:

$$\text{Hue} = 255/2\pi \times \arctan2 (\tfrac{1}{2}\sqrt{3} \times (\text{Green}-\text{Blue}) , \text{Red} - (\text{Green}+\text{Blue}) / 2) \times 240/255$$
$$\text{Saturation} = \sqrt{} (\text{Red}^2 + \text{Green}^2 + \text{Blue}^2 - \text{Red}\times\text{Green} - \text{Red}\times\text{Blue} - \text{Green}\times\text{Blue}) \times 240/255$$
$$\text{Intensity} = 1/3 \times (\text{Red} + \text{Green} + \text{Blue}) \times 240/255$$

Red, green, and blue values range from 0 to 255. Hue, saturation, and intensity values however range from 0 to 240; this range complies with the Windows color scheme definition. In the formulas above, multiplication factor 240/255 is used to obtain that range.

References:

- Heckbert, P., 1982. Color image quantization for frame buffer display. SIGGRAPH '82 Proceedings, p. 297.

7.3.4 Resample

Functionality

The Resample operation resamples a raster map from the map's current georeference to another target georeference. The coordinate of each output pixel is used to calculate a new value from close-by pixel values in the input map. Three resampling methods are available: nearest neighbour, bilinear interpolation, and bicubic interpolation.

In raster operations (e.g. MapCalc, Cross), all input raster maps must have the same georeference. Thus, prior to such operations, use Resample:

- *to combine raster maps from various sources*, when maps use different coordinate systems (projections) or different georeferences (pixel size): resample the maps to one common georeference;
- *to combine satellite imagery of different dates or resolutions*: create a georef tiepoints for each set of images, then resample the images preferably to a georef corners;
- *to combine satellite images with rasterized vector maps*: rasterize the vector data on the georef tiepoints of the satellite images, or, in case you prefer North-oriented raster maps, rasterize the vector maps with a georef corners, and resample the images with the georef tiepoints to this georef corners;
- *to combine scanned photographs with rasterized vector data*, or *to rectify scanned aerial photographs*: create a georef tiepoints, a georef direct linear or a georef orthophoto for the photo, then resample the photo to a georef corners.

You will usually resample raster maps from their current georeference to one common georeference corners.

☞ It is not advisable to use Resample to make a raster map with any georeference use a georeference 3D; use the Apply 3D operation instead.

For more information on georeference types or on creating georeferences, refer to ILWIS objects: georeferences or How to create a georeference. For more information on editing a georef tiepoints, a georef direct linear or a georef orthophoto, refer to Tiepoint editor.

Resampling methods

To resample an image, select an input image which has a georeference (usually a georeference tiepoints), select a resampling method (nearest neighbour, bilinear, bicubic), type an output map name and select the target georeference (usually a georeference corners).

- When using nearest neighbour resampling, the value of the input pixel closest to a new output pixel is used as output value (see Fig. 1);
- when using bilinear resampling, the values of 4 input pixels closest to a new output pixel are used to interpolate output values (see Fig. 2);
- when using bicubic resampling, the values of 16 input pixels closest to a new output pixel are used to interpolate output values.

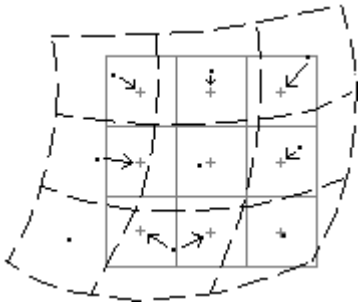


Figure 1: Nearest neighbour resampling. Dashed black lines represent Input pixels, black dots represent the coordinates of input pixels; the rectangular grid represents the output pixels, the plus marks indicate the coordinates of output pixels. The arrows indicate how output values are determined.

As you can see in Figure 1, some values of the input map may be used twice in the output map, while other input values may not be used at all.

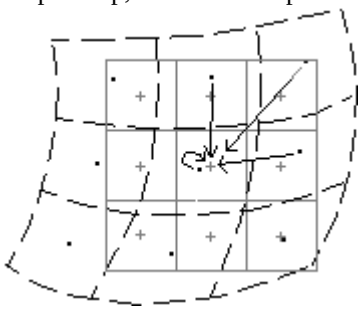


Figure 2: Bilinear resampling. Dashed black lines represent Input pixels, black dots represent the coordinates of input pixels; the rectangular grid represents the output pixels, the plus marks indicate the coordinates of output pixels. The arrows indicate how output values are determined.

Input map requirements
The input map needs to have a georeference; this is usually a georef tiepoints, a georef direct linear or a georef orthophoto.
For nearest neighbour resampling, the input map can have any domain.
For bilinear and bicubic resampling, the input map needs to be a value map.

Domain and georeference of output map
The output map uses the same domain as the input map. If the input map has a value domain, the value range and precision can be adjusted for the output map.

The target georeference for the output map has to be selected or created. You can usually select an existing georeference corners.

Dialog box

Dialog box options

Input raster map:	Select an input raster map. Open the list box and select the desired input map, or drag a raster map directly from the Catalog into this box. The input raster map may not use georef `None`.
Resampling method:	Select a resampling method: Nearest Neighbour, Bilinear, or Bicubic.
Output raster map:	Type a name for the output raster map that will contain resampled pixels.
Georeference:	Select an existing target georeference for the output raster map to which the input map should be resampled; open the list box by clicking it. Or create a new georeference by clicking the little create button.
Value range:	In case the output map uses a value domain: accept the default value range, or specify your own range of possible values in the output map.
Precision:	Accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Resample operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapResample`(InputMapName, Georeference, `NearestNeighbour` | `BiLinear` | `BiCubic` [, `Patch` | `NoPatch`)]

where:

OUTMAP	is the name of your output raster map.
`MapResample`	is the command to start the Resample operation.
InputMapName	is the name of your input raster map.
Georeference	is the name of an existing target georeference that should be used for the output raster map.
`NearestNeighbour`	is the parameter for Nearest Neighbour resampling.

`BiLinear`	is the parameter for BiLinear resampling.
`BiCubic`	is the parameter for BiCubic resampling.
`Patch`	is an optional parameter to have the operation first patch the input raster map, then resample it and finally unpatch it. This is the default behaviour.
`NoPatch`	is an optional parameter to have the operation directly resample the input raster map without patching. This will take less disk space but usually will take more time.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Resample operation resamples a raster map from the map's current georeference to another target georeference. The coordinate of each output pixel is used to calculate a new value from close-by pixel values in the input map. Three resampling methods are available: nearest neighbour, bilinear interpolation, and bicubic interpolation.

The resampling process consists of several steps:

- the selected output georeference determines the number of rows and columns in the output map; thus the XY-coordinate for each output pixel is known;
- next these positions are looked up in the original map and, according to the selected interpolation method, 1 (nearest neighbour), 4 (bilinear) or 16 (bicubic) neighbour pixels around this position in the input map are used to calculate a value for the output map.

Nearest neighbour resampling is the fastest method, but results in discontinuities because some input values may be used more than once as output value, while other input values may not be used at all. Bilinear resampling takes much less time than a bicubic resampling. A bilinear interpolation results in discontinuity of the first derivative. A bicubic interpolation remains continuous up to the second derivative.

Nearest neighbour resampling
With nearest neighbour resampling, first the coordinate of each pixel in the output map is determined. Then, for each output pixel, the pixel value of input pixel closest to this coordinate is used as output value.

Figure 1 below shows the position of a 'new' pixel in the output map, and the position and values of 4 surrounding pixels in the input map.

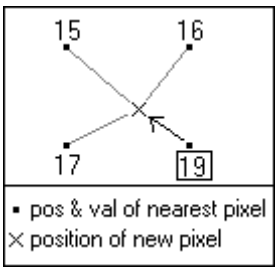


Figure 1: Nearest Neighbour resampling. Dots represent the coordinates of the input pixels; the cross indicates the coordinate of an output pixel. The arrow shows how the value of the nearest input pixel is assigned to the output pixel.

The value for the 'new' pixel in the output map is the value in the input map closest to the new coordinate (19).

Bilinear resampling
First the coordinate of each pixel in the output map is determined. Then the values of 4 surrounding pixels of the input map are used to calculate an interpolated value for each pixel in the output map.

Figure 2 below shows the position of a 'new' pixel in the output map, and the position and values of 4 surrounding pixels in the input map.

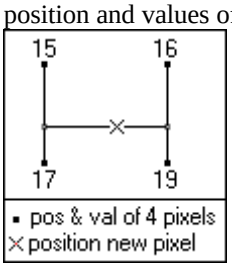


Figure 2: Bilinear interpolation in Bilinear Resampling. The numbered dots represent the coordinates of 4 neighbouring input pixels. The cross represents the coordinate of an output pixel. The straight lines represent the interpolations; intermediate answers are indicated by the unnumbered dots.

The value of the 'new' pixel in the output map is calculated by:
- first 2 interpolations in y-direction (between values 15 and 17, and between values 16 and 19) resulting in two intermediate values which are unnumbered
- then 1 interpolation in x-direction (between the two intermediate values).

A straight line is drawn through each set of 2 points, and from this the value of the third point is known. A bilinear interpolation should not be used when you intend to calculate a derivative of the output map.

Bicubic resampling
With bicubic resampling, first the coordinate of each pixel in the output map is determined; then the values of 16 surrounding pixels of the input map are used to calculate an interpolated value for each pixel in the output map.

Figure 3 below shows the position of a 'new' pixel in the output map, and the position and values of 16 surrounding pixels in the input map.

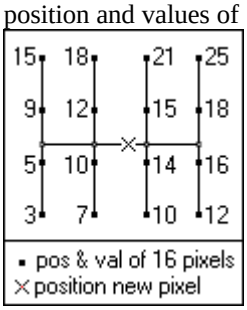


Figure 3: Bicubic interpolation in Bicubic resampling. The numbered dots represent the coordinates of 16 neighbouring input pixels. The cross represents the coordinate of an output pixel. The straight black lines represent the interpolations; intermediate answers are represented by unnumbered dots.

The value of the 'new' pixel in the output map is calculated by:
- first 4 interpolations in y-direction,
- then 1 interpolation in x-direction (between the 4 intermediate values).

A third order polynomial is fitted through each set of 4 known points and from this the value of the fifth point is known. A bicubic interpolation gives a better estimate of the output value than a bilinear interpolation.

7.4 Statistics

7.4.1 Optimum Index Factor

Functionality / Algorithm

The Optimum Index Factor (OIF) is a statistic value that can be used to select the optimum combination of three bands in a satellite image with which you want to create a color composite. The optimum combination of bands out of all possible 3-band combinations is the one with the highest amount of 'information' (= highest sum of standard deviations), with the least amount of duplication (lowest correlation among band pairs).

Preparation
- Create a map list containing the multi-spectral bands of your satellite image.
- Calculate a variance-covariance matrix or a correlation matrix of this map list.
- Open the Properties dialog box of your map list and click the Additional Info button.

The ranked OIF values are shown with the corresponding band combinations. For more information, refer to How to calculate Optimum Index Factor.

Example
Consider an input map list containing 7 bands, named `tmb1`, `tmb2`, `... tmb7`. For each combination of three bands in the map list, OIF values are calculated through a

simple formula which uses the standard deviations of the bands and correlation coefficients between band pairs (see algorithm below). The OIF values may read:

```
OIF Index Highest Ranking
1:  tmb4    tmb5    tmb6     ( 29.04)
2:  tmb1    tmb5    tmb6     ( 28.58)
3:  tmb3    tmb5    tmb6     ( 27.98)
4:  tmb5    tmb6    tmb7     ( 26.67)
5:  tmb1    tmb4    tmb5     ( 26.42)
6:  tmb2    tmb5    tmb6     ( 26.01)
```

The OIF values suggest that from the 7 bands in the map list, the combination of bands `tmb4`, `tmb5` and `tmb6` is the best statistical choice to create a color composite.

Notes

- By using the three bands with the highest OIF value for a color composite, it is not implied that you will create the 'best' color composite since this greatly depends on the purpose of your work.
- 'Noise' (such as dropouts) in one of the input bands is considered as high variance, hence this band will appear in all high ranking combinations.

Input requirements

To calculate Optimum Index Factors, a map list is required which contains at least 3 raster maps; the raster maps must all use the `Image` domain or the same value domain, and they must have the same georeference. Furthermore, it is necessary that a correlation matrix or a variance-covariance matrix has been calculated for the map list; this will provide the standard deviations and correlation coefficients, which are required for the OIF calculation.
For more information, refer to How to calculate Optimum Index Factor.

Output OIF values

After a variance-covariance matrix or a correlation matrix has been calculated for the input map list, you can display the ranked OIF values and corresponding band combinations by clicking the Additional Info button in the Properties dialog box of the input map list.
The OIF values are stored in the object definition file of the map list (.MPL).

Note: It is not possible to calculate OIF values from the command line.

Algorithm

1. First the number of possible combinations of three bands within the map list is determined as:

$$\binom{N}{3} = \frac{N!}{(3!(N-3)!)}$$

where:
N is the total number of bands in the map list.

For 3 bands, there is only 1 combination;
for 4 bands, there are 4 combinations;
for 5 bands, there are 10 combinations;
for 6 bands there are 20 combinations; and
for 7 bands, there are 35 combinations.

2. Then, for each combination of three bands, the OIF is calculated as:

$$OIF = \frac{Std_i + Std_j + Std_k}{|Corr_{i,j}| + |Corr_{i,k}| + |Corr_{j,k}|}$$

where:

Std_i	standard deviation of band i
Std_j	standard deviation of band j
Std_k	standard deviation of band k
$Corr_{ij}$	correlation coefficient of band i and band j
$Corr_{ik}$	correlation coefficient of band i and band k
$Corr_{jk}$	correlation coefficient of band j and band k

3. Finally, the OIF values are ranked.

7.4.2 Point statistics

Point statistics may help to get an impression of the nature of your point data prior to for instance a point interpolation, and to find necessary input parameters for kriging.

For point statistics a point map is required in which:
- the points themselves are values (point map with a value domain), for instance concentration values, or
- the points have an identifier (point map with a Class or ID domain) and values are stored in a column of the attribute table that is linked to the map.

Point statistics creates tables. By displaying the output tables, you can also display graphs.

Spatial correlation: calculates **spatial autocorrelation** (as Moran's I), **spatial variance** (as Geary's c) and semi-variance for point values that are at certain distances towards each other in a point map.
Regarding spatial autocorrelation and spatial variance: the user is encouraged to compare his or her data set with a second data set of the same point locations, but with a set of randomly generated attribute values, approximately in the same range as the measured variable. (Refer to the RND functions in Table Calculation). If the correlation/variance graphs are very much the same for the measured data and the random data, no autocorrelation exists between the data points. Hence, point interpolation is not useful.

By calculating semi-variances, you can display a semi-variogram. By modelling the semi-variogram, you can find the necessary input parameters, such as a model (spherical, exponential, etc.) and sill, range, and nugget values, for a kriging operation.

The Pattern analysis operation is a tool to obtain information on the spatial distribution of points in a point map. The output table contains six columns with the probabilities of finding 1 point (`Prob1Pnt`) within a certain distance from any point in your input map, then 2 points (`Prob2Pnt`), 3 points (`Prob3Pnt`), etc. Another column (
`ProbAllPnt`) contains the sum of `Prob1Pnt`, `Prob2Pnt`, … `Prob(n-1)`, in which n is the number of points in the input map.

By inspecting the graphs of distances against probabilities, you may recognize distribution patterns of your points like random, clustered, regular, paired etc.

7.4.3 Spatial correlation

Functionality

Spatial autocorrelation measures dependence among nearby values in a spatial distribution. Variables may be correlated because they are affected by similar processes, or phenomena, that extend over a larger region. Odland (1988, p.7) mentions that spatial autocorrelation 'exists whenever a variable exhibits a regular pattern over space in which values at a certain set of locations depend on values of the same variable at other locations'.

For example, if the concentration of a certain pollutant is very high at a certain location, it will most likely also be high in the direct surroundings. In other words, the concentration is autocorrelated at small distances. At larger distances, it is less likely that the concentration will be equally high. The correlation will probably be lower, and the variance higher.

By plotting the answers on autocorrelation against the distance classes, you will be able to see until which distance spatial autocorrelation exists between point pairs. This value can be used for the limiting distance in point interpolations such as moving average and moving surface. Furthermore, the user is encouraged to compare his or her data set with a data set consisting of the same point locations, with a set of attribute values, approximately in the same range as the measured variable, but created at random (using one of the RND functions in Table Calculation). If the graphs are very much the same for the measured data and the random data, no spatial autocorrelation exists between the data points. Hence, point interpolation is not useful.

Calculation of the semi-variance is a basic geostatistical measure to determine the rate of change of a regionalized variable along a specific orientation (usually distances). Semi-variance is defined as the sum of the squared differences between pairs of points separated by a certain distance divided by two times the number of points in this distance class. In a graph, semi-variance results can be plotted against distances; this is known as a semi-variogram. By modeling the semi-variogram, you can obtain necessary input information (such as model type, sill, range, and nugget)

for a Kriging operation later on. For more information, see the Additional information on semi-variograms below.

General process of this operation

1. First, the distances between all points are calculated.
2. Then, distance classes are determined. This is usually done according to the user-specified lag spacing: in the output table, records will appear for each multiple of the user-specified lag spacing, thus when using a lag spacing of 500 m., the distance values in the output table will be 0, 500, 1000, 1500, etc. However, these distance values in the output table represent the middle value of a distance class, thus for lag spacing 500, distance 500 represents the distance interval of 250-750m, distance 1000 represents the distance interval of 750-1250m, etc.
   When a variable was sampled at regular distances, you can use this distance for the lag spacing.
   On the command line, you can also use a certain expression to obtain log-scaled distance classes.
3. Subsequently, for each distance class, the number of point pairs is counted of which the points have such a distance towards each other.
   Thus, when the user-specified lag spacing is 500 m.:
   - the first record in the output table has value 0 in column Distance; this first distance class is only half a distance class: it contains all point pairs of which the distance of the points towards each other is 0-250 m.;
   - the second record in the output table has value 500 in column Distance; this distance class contains all point pairs of which the distance of the points towards each other is between 250-750 m.;
   - the third record in the output table has value 1000 in column Distance; this distance class contains all point pairs of which the distance of the points towards each other is between 750-1250 m. etc.;
4. Then, for all the point pairs within a certain distance class, the following statistical values are calculated:
   - spatial autocorrelation (as Moran's I)
   - spatial variance (as Geary's c)
   - semi-variance.

The formula to calculate semi-variance reads:

$$\gamma = \sum (z_i - z_{i+h})^2 / 2n$$

Where:

γ	semi-variance of points that have a certain distance (h) towards each other
z_i	the value of point i
z_{i+h}	the value of a point at distance h from point i
$\sum(z_i - z_{i+h})^2$	the sum of the squared differences between point values of all point pairs that have distance h towards each other
n	the number of point pairs within a distance class

Methods

In the dialog box, you can choose to use either the omnidirectional or the bidirectional method:

- The omnidirectional method simply determines all distances between all point pairs, regardless of any direction, i.e. in all directions. Thus, all point pairs that have a certain distance towards each other will be counted in a certain distance class. Then, Moran's I, Geary's c, and the semi-variance are calculated for all point pairs within each distance class.
- The bidirectional method first counts, just like the omnidirectional method, all pairs of points that have a certain distance to each other, and then calculates the Moran's I and Geary's c for these point pairs within each distance class. Furthermore, all point pairs are counted with a certain distance to each other and with a certain direction towards each other. For these point pairs, semi-variance will be calculated. Then, also, for the direction perpendicular to the specified direction, point pairs are counted and semi-variances calculated.

Both for the omnidirectional or the bidirectional method, linear distance intervals are created where the upper limits of these distance classes are multiples of the user-specified lag spacing.

To calculate semi-variances in a certain direction, you thus have to use the bidirectional method. You will have to supply two parameters: a direction angle and a tolerance angle. When you use a direction angle of 90°, it means that only point pairs for which the points are located in West-East or in East-West direction will be considered (i.e. +90° clockwise from the Y-axis). When using a tolerance of 10°, the direction of every 2 points may differ -10° or +10° from the specified direction (90°). So, in fact, all points that are found in a position within 80° to 100° to one another are valid pairs. Then, for the valid point pairs, the distance class to which they belong will be determined.

Optionally, you can specify a third parameter, the band width (m), to limit the tolerance angle to a certain width.
The parameters for the bidirectional method are schematically presented in Figure 1.

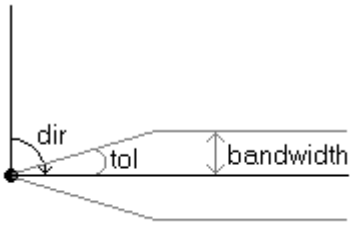


Figure 1: Schematic explanation of the parameters for the bidirectional method with which semi-variances will be calculated for the specified direction as well as for the perpendicular direction. The user has to specify a direction (Dir, the black angle) and a tolerance (Tol, the gray angle), and optionally, also a band width (the gray distance in meters) can be specified. These parameters are used to find valid point pairs. When an input point is located at the origin of this picture, it is calculated if any other input point is within the specified direction, tolerance angle and bandwidth. If this is the case, the 2 points are a valid point pair; otherwise the pair is ignored. For each valid point pair, the distance between the 2 points is calculated, and the point pair is counted in the appropriate distance class.

Finally, from the command line, you can even use another method by which logarithmic distance intervals are used. The lag spacing increases with the distance.

Input map requirements
The input point map should either be a value map itself, or a Class or ID point map which has a linked attribute table with one or more value columns.

Output table
An output table with domain None is created.

When you use the option Omnidirectional, the output table will contain 5 columns:
- Column `Distance` lists the middle values of the distance intervals;
- Column `NrPairs` lists for each distance interval, the number of point pairs found at these distances towards each other;
- Column `I` lists for each distance interval, the spatial autocorrelation of the point pairs in this distance interval;
- Column `c` lists for each distance interval, a statistic for spatial variance of the point pairs in this distance interval;
- Column `Semivar` lists for each distance interval, the semi-variance of the point pairs in this distance interval.

When you use the option Bidirectional, the output table will contain 8 columns:
- Column `Distance` lists the middle values of the distance intervals;
- Column `NrPairs` lists for each distance interval, the number of point pairs found at these distances towards each other;
- Column `I` lists for each distance interval, the spatial autocorrelation of the point pairs in this distance interval;
- Column `c` lists for each distance interval, a statistic for spatial variance of the point pairs in this distance interval;
- Column `NrPairs1` lists for each distance interval, the number of point pairs found in the user-specified direction and at these distances towards each other;
- Column `Semivar1` lists for each distance interval, the semi-variance for the point pairs found in the user-specified direction and at these distances towards each other;
- Column `NrPairs2` lists for each distance interval, the number of point pairs found perpendicular to the user-specified direction and at these distances towards each other;
- Column `Semivar2` lists for each distance interval, the semi-variance for the point pairs found perpendicular to the user-specified direction and at these distances towards each other.

Additional information: Semi-variograms

From the results of this operation, you can make a semi-variogram. A semi-variogram model describes the expected difference in value between pairs of samples with a given relative orientation.

- Display the output table of the Spatial correlation operation in a table window.
- Inspect the output table:
  - it is advised that the distance classes contain at least 30 point pairs, otherwise the calculated values will not be very reliable;
  - usually not more than half of the total sampled distance should be taken into account; the larger the distance between the point pairs, the less point pairs, the less reliable the outcome;
  - determine the variance (σ^2) of your variable in the input table, or in a histogram of the input map. The variance of a column can be calculated by using an expression like `OUT = var(`*`columnname`*`)`
- From the Options menu in the Table window, choose the Show Graph command.
- In the Graph dialog box, choose for the X-axis: the `Distance` column; and choose for the Y-axis: the `SemiVar` column.
  In case you used the bidirectional method, you can draw two graphs:
  - one with column `SemiVar1` as the Y-axis (this is the semi-variance in the direction which you specified) and
  - one with column `SemiVar2` as the Y-axis (this is the semi-variance in a direction perpendicular to the direction which you specified).
- In the Edit Graph dialog box, choose Points to represent the semi-variance values as points.

In literature, the shown graph is called a discrete experimental semi-variogram.

Ideally, a semi-variogram has the shape of Figure 2:

- When the distances between sample points is 0, the differences between sampled values is also expected to be 0. Thus, the semi-variance at distance 0 is 0, and when a line plotted through the points, this line will pass through the origin of the graph.
- Samples that are at a very small distance to each other are expected to have almost the same values; thus, the squared differences between sample values are expected to be small positive values at small distances.
- With increasing distance between point pairs, the expected squared differences between point values will also increase.
- At some distance the points that are compared are so far apart that they are not any more related to each other, i.e. the sample values will become independent of one another. Then, the squared differences of the point values will become equal in magnitude to the variance of the variable. The semi-variance no longer increases and the semi-variogram develops a flat region, called the sill. The distance at which the semi-variance approaches the variance is referred to as the range or the span of the variable.

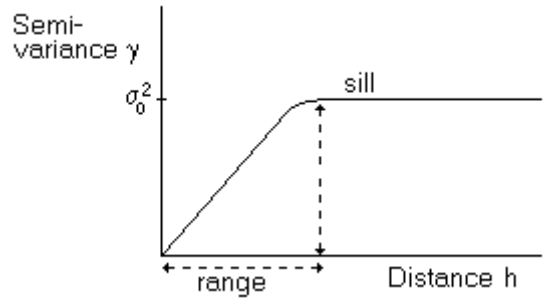


Figure 2: Ideal shape of a semi-variogram.

Remarks on semi-variograms:

- A semi-variogram with a nugget effect is a semi-variogram, which does not go through the origin. The variable is so erratic over very short distances that the semi-variance goes from zero to the level of the nugget effect in a distance less than the sampling distance: i.e. the variable is highly variable over distances less than the specified lag spacing or the sampling interval.
- For semi-variograms, which, after a flat sill level, show an ongoing increase in semi-variance values, probably a trend has to be taken into account for the longer distances. However, the semi-variance values for the distances up to the sill, are probably accurate enough to be used in a model.
- Possible dips in the semi-variogram indicate that at certain distances between points there is less difference between the samples than at other distances; this might indicate periodic trends.

The next step, before Kriging, is to model the discrete values of your experimental semi-variogram by a continuous function, which will give an expected value for any desired distance.

- From the Graph menu, choose the Add Semi-variogram Model command.
- In the Add Graph Semi-variogram dialog box, you can choose a type of semi-variogram model (spherical, exponential, etc.), and you can fill out values for the sill, range and nugget. Next a line will be drawn through your semi-variance values according to the model you selected and the values you selected for sill, range and nugget. You are advised to visually experiment a little with models and sill, range, and nugget values to find the best line through the experimental semi-variance values.

For more information on drawing lines through a semi-variogram, refer to the Graph window: Add Semi-variogram Model dialog box. Once, you have decided which model, and which values for sill, range and nugget fit your data best, you can continue with the Kriging operation.

References

- Clark, I. 1979. Practical geostatistics. Applied Science Publishers, London. 129 pp.
- Davis, J. C. 1973. Statistics and data analysis in geology. Wiley, New York. 646 pp.

- Isaaks, E. H., and R. M. Srivastava. 1989. An introduction to applied geostatistics. Oxford University Press, New York. 561 pp.
- Odland, J. 1988. Spatial autocorrelation. In: G.I. Thrall (Ed.), Sage University Scientific Geography Series no. 9. Sage Publications, Beverly Hills. 87 pp.

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. You can select a point map with a value domain, or a point map with a class or ID domain, which has a linked attribute table with values.
Column:	In case you selected an input point map with a class or ID domain, select an attribute column (value domain) from the attribute table that is linked to the map.
Omnidirectional:	Choose the option omnidirectional when distances between points should be calculated in all directions. Depending on the user-specified lag spacing, distance classes are determined; the upper limits of the distance classes are multiples of the lag spacing. Based on the found distance between each pair of points, it is determined to which distance class this pair of points belongs. Thus, all point pairs that are found to have a certain distance towards each other, will be counted as a point pair in that distance class. For all point pairs within each distance class, the spatial autocorrelation, spatial variance and semi-variance will be calculated.
Bidirectional:	Choose the option bidirectional when distances between points should also be calculated in a certain direction. Like the Omnidirectional option, all points that have a certain distance to each other will be counted as a point pair in a certain distance class. For all point pairs within each distance class, the spatial autocorrelation and spatial variance will be calculated. Furthermore, for the same distance classes, all points that have a certain distance to each other and that fall within the specified direction, will be counted as a point pair. For all these point pairs, the semi-variance will be calculated. Similarly, point pairs will be counted in the perpendicular direction and semi-variances will also be calculated for this perpendicular direction.
Lag spacing (m):	Type a value that will be used for the distance classes; the output table will contain a Distance column with multiples of the specified lag spacing. However, these values represent the middle value of a distance class. For example, when using lag spacing 500, distance classes of 500 m will be used, and the output table will show Distance values of 0, 500, 1000, 1500,

	etc. These values represent the intervals 0-250, 250-750, 750-1250, 1250-1750, etc. - For distance class 0, all point pairs will be counted where the points are less than 250 m apart; - for distance class 500, all point pairs will be counted where the points are between 250 and 750 m apart; - for distance class 1000, all point pairs will be counted where the points are between 750 and 1250 m apart, etc. When a variable was sampled at regular distances, you can enter this distance as the lag spacing. You should not use a value larger than half the size of your map.
Direction (deg):	For Bidirectional: type a value for the direction in which point pairs should be found. The direction is specified in degrees as a clockwise angle from the Y-axis. Direction 0° means finding only point pairs in the direction North-South and South-North to each other. Direction 90° means finding only point pairs in the direction East-West and West-East to each other. $0° \leq \text{direction} \leq 90°$.
Tolerance (deg):	For Bidirectional: type a value in degrees for half of the opening angle with which point pairs should be found in a certain direction. Tolerance 0.01 means a very narrow opening to find point pairs in a certain direction. Tolerance 45° means an opening of 90°. $0° < \text{tolerance} \leq 45°$.
Band width (m):	For Bidirectional: optionally, specify a value for the bandwidth. By specifying a bandwidth, you will limit the opening angle to a certain width; the opening width with which point pairs can be found in a certain direction will never be broader than twice the specified bandwidth.
Output table:	Type a name for the output table that will contain the spatial autocorrelation, spatial variance, and semi-variance.
Show:	Select this check box if you want the output table to be displayed in a table window when the operation has finished. Clear this check box if you do not want to see this table immediately: you simply define how the output table should be created.
Description:	Optionally, type a description for the output table. The description appears in the title bar when the table is displayed.

A dependent output table is created.

Command line

The Spatial correlation operation be directly executed by typing one of the following expressions on the command line of the Main window:

OUTTABLE = `TableSpatCorr`(InputPointMap)

OUTTABLE = `TableSpatCorr`(InputPointMap, *LagSpacing*)
OUTTABLE = `TableSpatCorr`(InputPointMap, *LagSpacing*, *Direction* [, *Tolerance* [, *Bandwidth*]])

where:

OUTTABLE	is the name of your output spatial correlation table.
`TableSpatCorr`	is the command to start the Spatial correlation operation.
InputPointMap	is the name of your input point map (value map). When you want to use a Class or ID point map with an attribute table linked to the map, you can use: *InputPointMap.ColumnName*
LagSpacing	is a parameter to specify the length in meters of the linear distance intervals that should be used. When this parameter is not used, logarithmic distance intervals will be calculated.
Direction	is a parameter to specify an angle in degrees for the direction in which point pairs should be found. Direction 0° is North; direction 90° is East. The direction is a clockwise angle from the Y-axis. 0° ≤ direction ≤ 90°. By using a direction, you will obtain bidirectional semi-variance results. When this parameter is not used your semi-variance results will be omnidirectional.
Tolerance	is a parameter to specify half of the opening angle in degrees with which point pairs in a certain direction should be found. 0° < tolerance ≤ 45°. When a Direction is specified but the Tolerance is not specified, a Tolerance of 45° will be used (meaning that all points will be found).
Bandwidth	is a parameter to specify half of the maximum width in meters within which point pairs within a certain angle and in a certain direction should be found. By specifying a bandwidth, you will limit the opening angle to a certain width; the opening width with which point pairs can be found in a certain direction will never be broader than twice the specified bandwidth. When this parameter is not specified, there is no limitation for the width of the opening angle.

When the first formula is used, only spatial autocorrelation and spatial variance are calculated, i.e. no semi-variances. Furthermore, instead of linear distance classes, you will obtain a number of logarithmic distance classes.

When the definition symbol = is used, a dependent output table is created; when the assignment symbol := is used, the dependency link is immediately broken after the output table has been calculated.

Algorithm

First, distances between all points are calculated. Distance classes are created for point pairs that are more or less at the same distance to each other. Distance classes are usually based on a user-specified lag spacing.

Then, for all point pairs within a distance group, the spatial autocorrelation, spatial variance and semi-variance is calculated.

In ILWIS, spatial autocorrelation between points is calculated as Moran's I (Odland):

$$I = \frac{n}{\sum\sum w_{ij}} \frac{\sum\sum w_{ij}(z_i - \bar{z})(z_j - \bar{z})}{\sum(z_i - \bar{z})^2}$$

In ILWIS, the spatial variance is calculated as Geary's c (Odland):

$$c = \frac{n-1}{2 \cdot \sum\sum w_{ij}} \frac{\sum\sum w_{ij}(z_i - z_j)^2}{\sum(z_i - \bar{z})^2}$$

Semi-variance is defined as:

$$g = \frac{\sum\sum w_{ij}(z_i - z_j)^2}{2 \cdot \sum\sum w_{ij}}$$

where:

z	the value of a point
$\bar{z}$	the average value of all available point values
$(z_i - \bar{z})(z_j - \bar{z})$	the product of: the difference of the value of point i and the average value of all points, and the difference of the value of point j and the average value of all points
$(z_i - z_j)^2$	the squared difference of the values of points `i` and `j`; this is calculated for all point pairs within a distance class and summed.
$(z_i - \bar{z})^2$	the squared difference of the value of point i and the average value of all points; this is calculated for all points and then summed; this is a constant value (variance).
n	the total number of points
w_{ij}	weight of a point pair. When using the omnidirectional method, $w_{ij} = 1$ when a point pair belongs to a certain distance class, otherwise $w_{ij} = 0$. When using the bidirectional method, $w_{ij} = 1$ when a point pair belongs to a certain distance class and when within the direction, tolerance and bandwidth as specified by the user (see also Figure 1 in Spatial correlation: functionality); otherwise $w_{ij} = 0$. In the numerators (top of a fraction) of these formulas, the weights assure that only the values of points that have a certain distance towards each other will be taken into account in the calculations for that distance class. In the denominators of these formulas (bottom of a fraction), i.e. in the standardization parts of the formulas, the weights count the number of valid point pairs within a distance class.

All summations are from i=1 to n and from j=i+1 to n, thus every point pair is counted only once.

References

- Geary, R.C., 1954. The contiguity ratio and statistical mapping.
- Moran, P.A.P., 1948. The interpretation of statistical maps.
- Odland, J. 1988. Spatial autocorrelation. In: G.I. Thrall (Ed.), Sage University Scientific Geography Series no. 9. Sage Publications, Beverly Hills. 87 pp.

7.5 Interpolation

7.5.1 Point interpolation

A point interpolation performs an interpolation on randomly distributed point values and returns regularly distributed point values. This is also known as gridding. In ILWIS, the output values are raster values.

In an ILWIS point interpolation, the input map is a point map in which:

- the points themselves are values (point map with a value domain), for instance concentration values, or
- the points are identifiers (point map with an Identifier domain) and values are stored in a column of an attribute table linked to the point map.

The output of a point interpolation is a raster map. For each pixel in the output map, a value is calculated by an interpolation on input point values.

Several point interpolation methods are available:

- Nearest point: assigns to pixels the value, identifier or class name of the nearest point, according to Euclidean distance. This method is also called Nearest Neighbour or Thiessen. The points in the input point map for the Nearest point operation do not need to be values necessarily; point maps (or attribute columns) with a class, ID or bool domain are also accepted.
- Moving average: assigns to pixels weighted averaged point values. The weight factors for the points are calculated by a user-specified weight function. Weights may for instance approximately equal the inverse distance to an output pixel. The weight function ensures that points close to an output pixel obtain larger weights than points, which are farther away. Furthermore, the weight functions are implemented in such a way that points which are farther away from an output pixel than a user-defined limiting distance obtain weight zero; this speeds up the calculation and prevents artifacts.
- Trend surface: calculates pixel values by fitting one surface through all point values in the map. The surface may be of the first order up to the sixth order. A trend surface may give a general impression of the data. Surface fitting is performed by a least squares fit. It might be a good idea to subtract the outcome of a trend surface from the original data, and calculate the residuals.
- Moving surface: calculates a pixel value by fitting a surface for each output pixel through weighted point values. The weight factors for the points are calculated

by a user-specified weight function. Weights may for instance approximately equal the inverse distance to an output pixel. The weight function ensures that points close to an output pixel obtain larger weights than points, which are farther away. Furthermore, the weight functions are implemented in such a way that points which are farther away from an output pixel than a user-defined limiting distance obtain weight zero; this speeds up the calculation and prevents artifacts. Surface fitting is performed by a least squares fit.

- Kriging: assigns to pixels weighted averaged points values, like the Moving Average operation. The weight factors in Kriging are determined by using a user-specified semi-variogram model (based on the output of the Spatial correlation operation), the distribution of input points, and are calculated in such a way that they minimize the estimation error in each output pixel. Two methods are available: Simple Kriging and Ordinary Kriging. Optionally, an error map can be obtained which contains the standard errors of the estimates. The errors are assumed to have e normal (Gauss) distribution. The technique is derived from the theory of regionalized variables (Krige, Matheron).

Preparations

- Point interpolations assume spatial randomness of the input points. To investigate whether your points are *randomly distributed*, or appear clustered, regular, or paired, etc., you can use the Pattern analysis operation prior to a point interpolation.
  In case your points are regularly distributed, e.g. as a regular grid, it is advised to directly rasterize the points with the Points to Raster operation. Use a georeference, which ensures that each output pixel contains one point and that the points are positioned at the center of the pixels. Further interpolation on the raster map values can be performed using the Densify operation or the Resample operation (bilinear or bicubic interpolation).
- Furthermore, point interpolations assume a certain degree of spatial correlation between the input point values. To investigate whether your point values are spatially correlated and until which distance from any point this correlation occurs, you can use the Spatial correlation operation prior to a point interpolation. The distance over which the data are correlated can be used as the maximum limiting distance in Moving average, Moving surface or Kriging. When there is no correlation between input point values, interpolation is senseless.

☞ When your point map contains values in which all extremes of your measured variable are present (e.g. for height values all mountaintops and valleys are measured), then using a Moving average point interpolation will probably be sufficient.

☞ It is possible that your point map does not contain all extreme values of a measured variable (e.g. you have soil samples and you have measured pH values as part of a large soil survey). In that case the advice is to use the Moving surface operation; the Moving average operation is not suitable. When you find extremes with the Moving surface operation, you might decide to go back to the field and measure the variable at the position of the extreme value; this will improve the results of a subsequent Moving surface operation.

- When you have relatively few points, it is advised to use a Trend surface operation.
- When you have rainfall data from a number of rainfall stations, first subtract for known patterns (e.g. height influence), then perform a Trend surface operation, and finally add the known patterns again to the output map.
- To check whether you have enough points within the limiting distance in a Moving average or a Moving surface point interpolation, you can perform the calculation again with a limiting distance increased by a factor 2. When you find profound differences in outcomes, you have chosen the limiting distance too small in the first calculation.
- When using a Moving average or a Moving surface point interpolation, it is for time efficiency reasons strongly advised to choose a rather large pixel size for the output map. Further interpolation on the raster map values can be performed using the Densify operation or the Resample operation (bilinear or bicubic interpolation).
- Instead of using Densify or Resample, you can also use online interpolation. In the Properties dialog box of the output raster map of an interpolation, i.e. of a value raster map, you can select the Interpolation check box. This means that the normal pixel value will only refer to the center of that pixel; elsewhere in any pixel, a value will be directly interpolated based on the values of 4 (bilinear) or 16 (bicubic) neighbouring pixels. The interpolated values are directly available in the raster map, e.g. by using left mouse information or in pixel information. Hence, the creation of an extra raster map with Densify or Resample is not needed. For more information, see Raster Map Properties (dialog box).

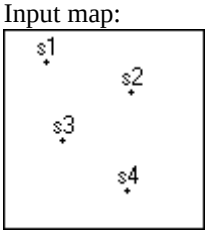
7.5.2 Nearest point

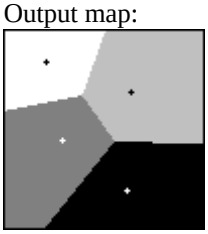
Functionality / Algorithm

The Nearest point operation requires a point map as input and returns a raster map as output. Each pixel in the output map is assigned the class name, identifier, or value of the nearest point.

Example
Points may represent schools, hospitals, water wells, etc. The output of a nearest point operation on such a point map gives the 'service area' of the schools, hospitals or water wells, based on the shortest distance (as the crow flies) between points and pixels.

Input map: Output map:





Nearest point operation versus Distance operation
The Nearest point is operation is also known as Nearest neighbour or Thiessen Map. Also the Distance operation has an option to create a Thiessen map.

- When every pixel in the output map is equally accessible, the Nearest point operation offers a quick way to obtain a Thiessen map from point data. Furthermore, as the Nearest point operation uses *Euclidean* distances, the output of the Nearest point operation may be somewhat more precise than the output of the Distance operation, which uses approximated raster distances.
- When you have many points or when you wish to use weights to indicate accessibilities, you can use the Distance operation. For more information, see Distance calculation: Thiessen map.

Input map requirements
No special input map requirements. Input maps may be maps of domain type class, ID, value, or bool. Furthermore, you can use a class or ID map with an attribute table.

Domain and georeference of output map
The output raster map uses the same domain as the input point map or the domain of the attribute column.
The georeference for the output map has to be selected or created; you can usually select an existing georeference corners.

Algorithm
For each output pixel, the Euclidean distances towards all points are determined. The value of the point with the shortest distance towards an output pixel is assigned to this output pixel.

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. Any map with a class, ID, value or bool domain is accepted.
Attribute:	Select this check box if you want to use an attribute column from the attribute table, which is linked to the input point map (class or ID map). Clear this check box to use the input point map.
Output raster map:	Type a name for the output raster map that will contain, for

	each pixel, the class name, identifier, or value of the nearest point.
Georeference:	Select the name of an existing georeference or create a new georeference (click the create button).
Value range:	In case the output map uses a value domain, accept the default value range, or specify your own range of possible values in the output map.
Precision:	In case the output map uses a value domain, accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

7.5.3 Moving average

Functionality

The Moving average operation is a point interpolation, which requires a point map as input and returns a raster map as output. The values for the output pixels are the weighted averages of input point values. Weighted averaging is the calculation of the sum of the products of weights and point values, divided by the sum of weights.

The weight factors for the input points are calculated by a user-specified weight function. There are two methods: inverse distance and linear decrease. Both methods ensure that points close to an output pixel obtain large weights and that points farther away from an output pixel obtain small weights. Values of points close to an output pixel are thus of greater importance to this output pixel value than the values of points farther away.

By specifying a limiting distance, you can influence until what distance from any output pixel, points will be taken into account for the calculation of a new value for that output pixel. For each output pixel, only the values of the points falling within the limiting distance to this output pixel will be used. Values of points that are farther away from an output pixel than the specified limiting distance, obtain weight zero by the weight calculation, and these values will thus not be used in the output pixel value calculation. This speeds up the calculation and prevents artifacts.

Input map requirements
The input point map should be a value map. Furthermore, when a point map uses a class or ID domain and the map is linked to an attribute table, you can also use such a point map and select a column with a value domain from the map's attribute table.

Domain and georeference of output map
The output raster map uses the same value domain as the input point map or the attribute column. The value range and precision can be adjusted for the output map. The georeference for the output map has to be selected or created; you can usually select an existing georeference corners.

- ☞ For time efficiency reasons, it is advisable to choose a rather large pixel size for the output raster map. Further interpolation on the raster values can be performed with the Densify operation or the Resample operation (using bilinear or bicubic interpolation).
- ☞ Prior to interpolation, you can use the Pattern analysis operation to investigate whether your points are randomly distributed, and the Spatial correlation operation to investigate whether your points are spatially correlated and until which distance from any point this correlation occurs. The limiting distance should not be specified larger than the distance until which correlation occurs.
- ☞ Make sure that there are enough points within the limiting distance; in other words, choose a limiting distance, which is large enough. To check whether you have enough points within the limiting distance, you can perform the calculation again with a limiting distance increased by a factor 2. When you find profound differences in outcomes, you have chosen the limiting distance too small in the first calculation.

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. You can select a point map with a value domain, or a point map with a class or ID domain, which has a linked attribute table with values.
Attribute:	In case you selected an input point map with a class or ID domain, select an attribute column (value domain) from the attribute table.
Weight function:	Select the weight function, which should be used to calculate weight factors for the points. Both weight functions ensure that points close to an output pixel will obtain a larger weight factor than points farther away. The inverse distance method assigns relatively larger weights to points close to an output pixel than the linear decrease method. For more information on weight functions, see Moving average: algorithm.

Inverse distance:	$(1 / d^n) - 1$
Linear decrease:	$1 – d^n$ d = relative distance of points towards pixels n = weight exponent
Weight exponent:	Type a value for weight exponent n to be used in the selected weight function (real value, usually a value around value 1.0).
Limiting distance:	Type a value for the limiting distance. Points that are farther away from any output pixel than the limiting distance are assigned weight zero; the values of these points will thus not be used in the calculation of the output value for that pixel.
Output raster map:	Type a name for the output raster map that will contain the weighted averaged point values.
Georeference:	Select the name of an existing georeference or create a new georeference. For time efficiency reasons, it is advisable to choose a rather large pixel size for the output raster map. Further interpolation on the raster values can be performed with the Densify operation or the Resample operation (using bilinear or bicubic interpolation).
Value range:	Accept the default value range, or specify your own range of possible values in the output map.
Precision:	Accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Moving Average operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapMovingAverage`(InputPointMap, Georeference, *WeightFunction*)
OUTMAP = `MapMovingAverage`(PointMap.Column, Georeference, *WeightFunction*)

where:

OUTMAP	is the name of the output raster map.
`MapMovingAverage`	is the command to start the Moving Average operation.

InputPointMap	is the name of the input point map with a value domain.
PointMap.Column	is the name of an input point map with a class or ID domain which has a linked attribute table, and the name of a value column in this attribute table.
Georeference	is the name of an exisiting georeference for the output raster map.
WeightFunction	is an expressing which defines the type of weight function to be used, as: `InvDist`(*Exp,LimDist*) `Linear`(*Exp,LimDist*)

where:

`InvDist`	is the parameter to indicate the use of the inverse distance method.
`Linear`	is the parameter to indicate the use of the linear decrease method.
Exp	is a value for weight exponent *n* to be used in the specified weight function (real value, usually a value close to 1.0).
LimDist	is a value for the limiting distance: points that are farther away from an output pixel than the limiting distance obtain weight zero.

Example
A full expression for the Moving average operation, using weights according to the inverse distance method, a weight exponent 1, and a limiting distance of 500 m. might thus read:

OUTMAP = `MapMovingAverage(`MapX, GeorefX, `InvDist(1,500))`

When the definition symbol `=` is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

Moving average performs a weighted averaging on point values and returns a raster map as output. The user has to specify a weight function and a limiting distance.

Step 1
For each output pixel, the distances of all points towards the output pixel are calculated to determine weight factors for the points:

For each output pixel, weight factors for the points are then calculated according to the weight function specified by the user. Two weight functions are available: inverse distance and linear decrease.

Inverse distance: weight = $(1 / d^n) - 1$
Linear decrease: weight = $1 - d^n$

where:
d = D/D_0 = relative distance of point to output pixel
D = Euclidean distance of point to output pixel
D_0 = limiting distance
n = weight exponent

Figures 1 and 2 below show the manner in which weight values decrease with increasing distance, for different values of n. The X-axes represent d: the distance of a point towards an output pixel divided by the limiting distance. The Y-axes represent the calculated **weight values**.

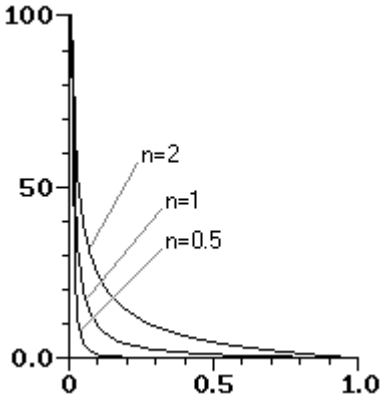


Figure 1: inverse distance
weight = $(1/d^n) – 1$
X-axis: $d = D/D_0$
Y-axis: weight values

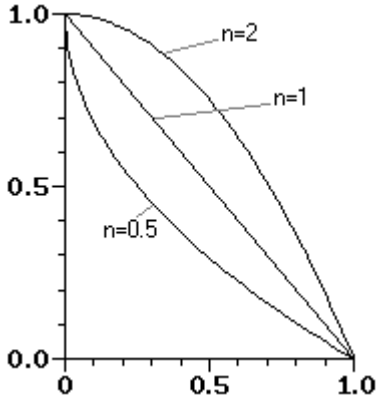


Figure 2: linear decrease
weight = $1 - d^n$
X-axis: $d = D/D_0$
Y-axis: weight values

The weight functions ensure that points close to an output pixel obtain a larger weight value than points, which are farther away from an output pixel.

See that when the distance of a point towards an output pixel equals the limiting distance (value 1.0 at X-axis), or when the distance of a point towards an output pixel is larger than the limiting distance, the calculated weight value will equal 0; the weight functions are thus continuous.

- The inverse distance function can be selected when you have very accurately measured point values and when local variation, within a pixel, is small. This function ensures that the computed output values equal the input point values.
- The linear decrease function can be selected for point maps in which you know there are measurement errors, and when points lying close to each other have different values. This function will decrease the overall error by correcting erroneous measurements with other close points. The consequence is that the

computed output values will not necessarily coincide with the measured point values.

Step 2
Then, for each output pixel, an output value is calculated as the sum of the products of calculated weight values and point values, divided by sum of weights.

output pixel value = $\sum(w_i \times val_i) / \sum w_i$

Where:
w_i = weight value for point i
val_i = point value of point i

☞ For time efficiency reasons, it is strongly advised to choose a rather large pixel size for the output raster map. Further interpolation on the raster values can be performed with the Densify operation or the Resample operation (using bilinear or bicubic interpolation).

7.5.4 Trend surface

Functionality

The Trend Surface operation is a point interpolation, which requires a point map as input and returns a raster map as output. One polynomial surface is calculated by a least squares fit so that the global surface approaches all point values in the map. The calculated surface values are assigned to the output pixels.

The trend surfaces in this operation range from a simple plane to complex polynomial surfaces. You can usually select a first or second order function to calculate the surface, as these are the least sensitive to produce extreme values. For more information on the available functions, which calculate surfaces, see Trend surface: algorithm.

Input map requirements
The input point map should be a value map. Furthermore, when a point map uses a class or ID domain and the map is linked to an attribute table, you can also use such a point map and select a column with a value domain from the map's attribute table.

Domain and georeference of output map
The output raster map uses the same value domain as the input point map or the attribute column. The value range and precision can be adjusted for the output map. The georeference for the output map has to be selected or created; you can usually select an existing georeference corners.

☞ After performing the Trend surface operation, you can press the Additional Info button in the Properties dialog box of the output raster map to display the formula of the calculated surface.

- ☞ You can subtract the output map of the trend surface operation from the original point values, e.g. using the MAPVALUE() and PNTCRD() functions in TabCalc. If the residuals of this subtraction do not have a strong spatial correlation, the trend surface is a representative model of your point values.
- ☞ Prior to interpolation, you can use the Pattern analysis operation to investigate whether your points are randomly distributed, and the Spatial correlation operation to investigate whether your points are spatially correlated.

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. You can select a point map with a value domain, or a point map with a class or ID domain, which has a linked attribute table with values.
Attribute:	In case you selected an input point map with a class or ID domain, select an attribute column (value domain) from the attribute table.
Surface:	Select one of the 8 functions to calculate a trend surface which approaches all points in your point map.
Output raster map:	Type a name for the output raster map.
Georeference:	Select the name of an existing georeference or create a new georeference.
Value range:	Accept the default value range, or specify your own range of possible values in the output map.
Precision:	Accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Trend surface operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapTrendSurface`(InputPointMap, Georeference, *SurfaceType*)
OUTMAP = `MapTrendSurface`(PointMap.Column, Georeference, *SurfaceType*)

where:

OUTMAP	is the name of the output raster map.
`MapTrendSurface`	is the command to start the Trend surface operation.
InputPointMap	is the name of the input point map with a value domain.
PointMap.Column	is the name of an input point map with a class or ID domain which has a *linked* attribute table, and the name of a value column in this attribute table.
Georeference	is the name of an existing georeference that should be used for the output raster map.
SurfaceType	is the parameter which specifies the function with which a surface should be calculated; use one of the following: `Plane` | `Linear2` | `Parabolic2` | `2` | `3` | `4` | `5` | `6`

When the definition symbol `=` is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Trend Surface operation is a point interpolation, which requires a point map as input and returns a raster map as output. One polynomial surface is calculated by a global least squares fit approaching all point values in the map. The calculated surface values are assigned to the output pixels.

Below the functions and surface types are listed, as well as the absolute minimum number of points that are mathematically required to fit such a surface. You will always need more points than this absolute mathematical minimum to obtain good results.

In general, the use of simple surfaces is preferred, as these will produce the least artificial extreme values.

Plane:	the surface is a plane, formula: $z = a + bx + cy$ Minimum number of points required: 3
2nd degree Linear:	the surface is planar but tilted, i.e. first order plane, formula: $z = a + bx + cy + dxy$ Minimum number of points required: 4
2nd degree Parabolic:	the surface is a second order polynomial surface, formula: $z = a + bx + cy + ex^2 + fy^2$ Minimum number of points required: 5
2nd degree:	the surface is a full second order polynomial surface, formula: $z = a + bx + cy + dxy + ex^2 + fy^2$ Minimum number of points required: 6

3rd degree:	the surface is a third order polynomial surface $z = a + ... + gx^3 + hx^2y + ixy^2 + jy^3$ Minimum number of points required: 10
4th degree:	the surface is a fourth order polynomial surface $z = a + ... + kx + lx^3y + mx^2y^2 + nxy^3 + oy$ Minimum number of points required: 15
5th degree:	the surface is a fifth order polynomial surface $z = a + ... +px + qxy +rx^3y^2 + ... + uy$ Minimum number of points required: 21
6th degree:	the surface is a sixth order polynomial surface $z = a + ... + vx+...$ Minimum number of points required: 28

7.5.5 Moving surface

Functionality

The Moving surface operation is a point interpolation, which requires a point map as input and returns a raster map as output. For each output pixel, a polynomial surface is calculated by a moving least squares fit; for each output pixel, the surface will approach the weighted point values of the points which fall within the specified limiting distance (see below). The calculated surface values are assigned to the output pixels.

The weight factors for the input points are calculated by a user-specified weight function. There are two methods: inverse distance and linear decrease. Both methods ensure that points close to an output pixel obtain large weights and that points farther away from an output pixel obtain small weights. The values of points close to an output pixel are thus of greater importance to the output value for that pixel than points farther away.

By specifying a limiting distance, you can influence until which distance from any output pixel, points will be taken into account for the calculation a value for that output pixel. For each output pixel, only the values of the points falling within the limiting distance to this output pixel will be used to calculate a value for that output pixel. Values of points that are farther away from an output pixel than the specified limiting distance, obtain weight zero by the weight calculation, and these values will thus not be used in the output pixel value calculation. This speeds up the calculation and prevents artifacts.

The surfaces in this operation range from a simple plane to complex polynomial surfaces. You can usually select a first or second order surface, as these are the least sensitive to produce artificial extreme values.

For more information on weight calculation methods and surfaces, see Moving surface: algorithm.

Input map requirements
The input point map should be a value map. Furthermore, when a point map uses a class or ID domain and the map is linked to an attribute table, you can also use such a point map and select a column with a value domain from the map's attribute table.

Domain and georeference of output map
The output raster map uses the same value domain as the input point map or the attribute column. The value range and precision can be adjusted for the output map. The georeference for the output map has to be selected or created; you can usually select an existing georeference corners.

- ☞ For time efficiency reasons, it is advisable to choose a rather large pixel size for the output raster map. Further interpolation on the raster values can be performed with the Densify operation or the Resample operation (using bicubic interpolation).
- ☞ Prior to interpolation, you can use the Pattern analysis operation to investigate whether your points are randomly distributed, and the Spatial correlation operation to investigate whether your points are spatially correlated and until which distance from any point this correlation occurs. The limiting distance should not be specified larger than the distance until which correlation occurs.
- ☞ Make sure that there are enough points within the limiting distance; in other words, choose the limiting distance value large enough.
  To check whether you have enough points within the limiting distance, you can perform the calculation again with a limiting distance increased by a factor 2. When you find profound differences in outcomes, you have chosen the limiting distance too small in the first calculation.
- ☞ To find only a regional trend, use the Trend surface operation, which is a relatively fast operation. When you use the residuals of the Trend surface operation as input for the Moving surface operation, you may be able to split regional and local phenomena.

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. You can select a point map with a value domain, or a point map with a class or ID domain, which has a linked attribute table with values.
Attribute:	In case you selected an input point map with a class or ID domain, select an attribute column (value domain) from the attribute table.

Weight function:	Select the weight function, which should be used to calculate weight factors for the points. Both weight functions ensure that points close to an output pixel will obtain a larger weight factor than points farther away. The inverse distance method assigns relatively larger weights to points close to an output pixel than the linear decrease method. For more information on weight functions, see Moving surface: algorithm.
Inverse distance:	$(1 / d^n) - 1$
Linear decrease:	$1 - d^n$ d = relative distance of points towards pixels n = weight exponent
Weight exponent:	Type a value for weight exponent *n* to be used in the selected weight function (real value, usually a value close to 1.0).
Limiting distance:	Type a value for the limiting distance. Points that are farther away from any output pixel than the limiting distance, are assigned weight zero; the values of these points will thus not be used in the calculation of the output value for that pixel.
Surface:	Select one of the 8 functions to calculate for each output pixel a surface which approaching all points within the limiting distance towards this output pixel.
Output raster map:	Type the name of the output raster map.
Georeference:	Type the name of an existing georeference or create a new georeference.
Value range:	Accept the default value range, or specify your own range of possible values in the output map.
Precision:	Accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Moving surface operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapMovingSurface`(InputPointMap, Georeference, *SurfaceType*, *WeightFunction*)
OUTMAP = `MapMovingSurface`(PointMap.Column, Georeference, *SurfaceType*, *WeightFunction*)

where:

OUTMAP	is the name of the output raster map.
`MapMovingSurface`	is the command to start the Moving surface operation.
InputPointMap	is the name of the input point map with a value domain.
PointMap.Column	is the name of an input point map with a class or ID domain which has a linked attribute table, and the name of a value column in this attribute table.
GeoReference	is the name of an existing georeference that should be used for the output raster map.
SurfaceType	is the parameter which specifies the function with which surfaces should be calculated; use one of the following: `Plane` | `Linear2` | `Parabolic2` | `2` | `3` | `4` | `5` | `6`
WeightFunction	is an expressing which defines the type of weight function to be used, as: `InvDist`(*Exp,LimDist*) `Linear`(*Exp,LimDist*)

where:

`InvDist`	is the parameter to indicate the use of the inverse distance method.
`Linear`	is the parameter to indicate the use of the linear decrease method.
Exp	is a value for weight exponent *n* to be used in the specified weight function (real value, usually a value close to 1.0).
LimDist	is a value for the limiting distance: points that are farther away from an output pixel than the limiting distance obtain weight zero.

Example
A full expression for the Moving surface operation, using a plane, using weights according to the inverse distance method, a weight exponent 1, and a limiting distance of 500m. might thus read:

OUTMAP = `MapMovingSurface`(MapX, GeoRefX, `Plane`, `InvDist`(1,500))

When the definition symbol `=` is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Moving surface operation is a point interpolation, which requires a point map as input and returns a raster map as output. For each output pixel, a polynomial surface is calculated by a least square method approaching weighted point values.

Step1
First, for each output pixel, the distances of all points towards the output pixel are calculated to determine weight factors for the points:

For each output pixel, weight factors for the points are then calculated according to the weight function specified by the user. Two weight functions are available: inverse distance and linear decrease.

Inverse distance: weight = $(1 / d^n) - 1$
Linear decrease: weight = $1 – d^n$

where:
d = D/D_0 = relative distance of point to output pixel
D = Euclidean distance of point to output pixel
D_0 = limiting distance
n = weight exponent

Figures 1 and 2 below show the manner in which weight values decrease with increasing distance, for different values of n. The X-axes represent d: the distance of a point towards an output pixel divided by the limiting distance. The Y-axes represent the calculated weight values.

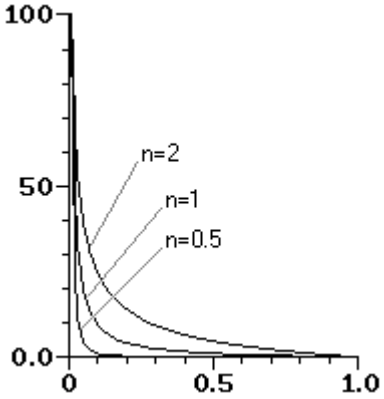


Figure 1: inverse distance
weight = $(1/d^n) - 1$
X-axis: $d = D/D_0$
Y-axis: weight values

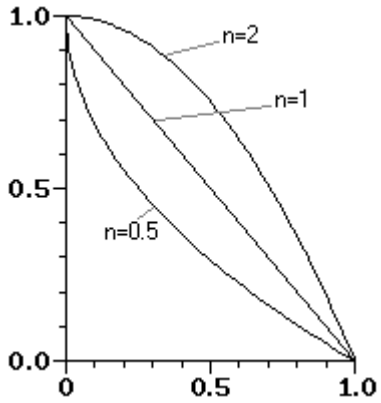


Figure 2: linear decrease
weight = $1 - d^n$
X-axis: $d = D/D_0$
Y-axis: weight values

The weight functions ensure that points close to an output pixel obtain a larger weight value than points farther away from an output pixel.

See that when the distance of a point towards an output pixel equals the limiting distance (value 1.0 at X-axis), or when the distance of a point towards an output pixel is larger than the limiting distance, the calculated weight value will equal 0; the weight functions are thus continuous.

☞ The *inverse distance* function can be selected when you have very accurately measured point values and when local variation, within a pixel, is small. This function ensures that the computed output values equal the input point values.

☞ The *linear decrease* function can be selected for point maps in which you know there are measurement errors, and when points close to each other have different values. This function will decrease the overall error by correcting erroneous measurements with other close points. The consequence is that the computed output values will not necessarily coincide with the measured point values.

Step 2
Next, for each output pixel, an output value is calculated by fitting a polynomial surface through all weighted point values of points that fall within the limiting distance towards this pixel. To each output pixel, the calculated surface value is assigned.

Below the functions and surface types are listed, as well as the absolute minimum number of points that are mathematically required to fit such a surface. To obtain good results, for each output pixel, you need more points within the limiting distance than this absolute mathematical minimum.

In general, the use of simple surfaces is preferred, as these will produce the least articial extreme values.

Plane:	the surface is a plane, formula: $z = a + bx + cy$ Minimum number of points required: 3
2nd degree Linear:	the surface is planar but tilted, i.e. first order plane, formula: $z = a + bx + cy + dxy$ Minimum number of points required: 4
2nd degree Parabolic:	the surface is a second order polynomial surface, formula: $z = a + bx + cy + ex^2 + fy^2$ Minimum number of points required: 5
2nd degree:	the surface is a full second order polynomial surface, formula: $z = a + bx + cy + dxy + ex^2 + fy^2$ Minimum number of points required: 6

3rd degree:	the surface is a third order polynomial surface $z = a + ... + gx^3 + hx^2y + ixy^2 + jy^3$ Minimum number of points required: 10
4th degree:	the surface is a fourth order polynomial surface $z = a + ... + kx + lx^3y + mx^2y^2 + nxy^3 + oy$ Minimum number of points required: 15
5th degree:	the surface is a fifth order polynomial surface $z = a + ... + px + qxy + rx^3y^2 + ... + uy$ Minimum number of points required: 21
6th degree:	the surface is a sixth order polynomial surface $z = a + ... + vx + ...$ Minimum number of points required: 28

☞ For time efficiency reasons, it is strongly advised to choose a rather large pixel size for the output raster map. Further interpolation on the raster values can be performed with the Densify operation or the Resample operation (using bicubic interpolation).

7.5.6 Kriging

Functionality

Kriging can be seen as a point interpolation, which requires a point map as input and returns a raster map with estimations and optionally an error map. The estimations are weighted averaged input point values, similar to the Moving Average operation. The weight factors in Kriging are determined by using a user-specified semi-variogram model (based on the output of the Spatial correlation operation), the distribution of input points, and are calculated in such a way that they minimize the estimation error in each output pixel. The estimated or predicted values are thus a linear combination of the input values and have a minimum estimation error. The optional error map contains the standard errors of the estimates.

Kriging is named after D.G. Krige, a South African mining engineer and pioneer in the application of statistical techniques to mine evaluation. The Kriging technique is derived from the theory of regionalized variables (Krige, Matheron). An advantage of Kriging (above other moving averages like inverse distance) is that it provides a measure of the probable error associated with the estimates.

Preparation
Besides an input point map, Kriging requires a semi-variogram model including values for the parameters nugget, sill and range; this can be obtained from the Spatial correlation operation.

- Display the output table of Spatial correlation, and create a graph (i.e. a semi-variogram) in which you plot the semi-variance values against the distance classes.
- Subsequently, model the semi-variogram: select a model like Spherical, Exponential, Gaussian, etc., and choose values for sill, nugget and range. This shows as a line in the graph through the points.
- Superimpose trials of various models with various defining parameters in order to find the best approximation.

For more information, see Spatial correlation : functionality, section on Semi-variograms, or the Graph window : Add semi-variogram model.

☞ If a trend is apparent, it should be removed before Kriging. One could use the Trend surface operation and appropriate TabCalc subtractions.

Methods

Two Kriging methods are available: Simple Kriging and Ordinary Kriging.

- In Simple Kriging, all input points are used to calculate each output pixel value. Only one large matrix needs to be inverted to find the weight factors for all input points.
- In Ordinary Kriging, you can influence the number of points that should be taken into account in the calculation of an output pixel value by specifying a limiting distance and a minimum and maximum number of points:
  - only the points that fall within the limiting distance to an output pixel will be used in the calculation for that output pixel value,
  - within the limiting distance towards each output pixel, at least the specified minimum number of points should be found, otherwise the pixel will be assigned the undefined value.
  - within the limiting distance towards each output pixel, only the specified maximum number of points will be taken into account; when more than the specified maximum number of points are found within the limiting distance, only the points which are nearest to the output pixel will be taken into account.

  For each output pixel, a set of simultaneous equations needs to be solved to find the weight values for those points that contribute to the output value of the pixel.

In general, it can be said that the more points are used, the more reliable the estimation will be.

Removing duplicates or coinciding points

When you have multiple values for the same location or when point locations are very close to each other, i.e. when samples coincide, the Kriging system of equations becomes singular and cannot be solved.
It is therefore advised to use the option Remove Duplicates, which will automatically 'remove' any coinciding points. You can choose to either take the average value of coinciding points, or to take the value of the first (coinciding) point encountered only. By specifying a Tolerance, you can control the distance between points at which points are considered coinciding or not. When the distance between

points is less than the specified tolerance, these points are considered coinciding; otherwise the points are considered distinct.

When your input data does contain coinciding points and when the Remove Duplicates option is not used, Simple Kriging will yield an error message, and Ordinary Kriging will assign the undefined value for all pixels to which the coinciding points make a contribution.

Error map
The error map contains the standard error of the estimate, i.e. the square root of the error variance.

The error variance in each estimated output pixel depends on:
- the semi-variogram model including its parameters,
- the spatial distribution of the input sample points,
- the position of an output pixel with respect to the position of the input sample points.

A standard error which is larger than the original sample standard deviation denotes a rather unreliable prediction.

Input map requirements
The input point map should be a value map. Furthermore, when a point map uses a ID domain and the map is linked to an attribute table, you can also use such a point map and select a column with a value domain from the map's attribute table. A current limitation of the operation is that Simple Kriging can only handle point maps with a maximum of 89 valid input points. Ordinary Kriging can only handle point maps that contain a maximum of 2500 input points. However, within each limiting distance (search radius) only a maximum of 89 valid points are allowed.

Domain and georeference of output raster map
The output raster map uses the same value domain as the input point map or the attribute column. The value range and precision can be adjusted for the output map; it is advised to choose a wider value range for the output map than the input value range. The georeference for the output map has to be selected or created; you can usually select an existing georeference corners.

☞ When the output raster map shows undefined pixels, this can be due to several factors:
- There are coinciding points while you did not use the option Remove Duplicates. Remedy: use the option Remove Duplicates, or change the tolerance. You can also remove duplicates yourself, e.g. by editing the point map in the point editor, by editing the point map as a table, or by editing the attribute table of the point map.
- The value range of the output raster map is too narrow: because of extrapolation, certain output values may fall beyond the range limits and ILWIS converts them

to undefined.
Remedy: extend the value range in the dialog box and compare the results.

- The minimum number of points to be taken into account is too high in relation to the limiting distance, i.e. the minimum number of points specified are not found within the specified limiting distance.
  Remedy: lower the minimum number of points and/or increase the limiting distance.

☞ The output may also become erratic when:

- The input points are too sparse in certain areas to ensure an estimate with small variance.
  Remedy: increase limiting distance, find supplementary point data, investigate possible anisotropy.
- The semi-variogram Range parameter is set incorrectly due to poor interpretation of a correct output of the Spatial correlation operation. This may occur when you chose the lag spacing in Spatial correlation far too large.
  Remedy: Try another lag spacing and inspect the Nr of pairs column.
- The semi-variogram model is incorrect: the geometric distribution of the sample points is unbalanced or the user is unaware of an existing anisotropy. There is perhaps an error in the range setting (horizontal scale in graph) or sill and/or nugget (vertical scale).

Dialog box

Dialog box options

Input point map:	Select an input point map. Open the list box and select the desired input map, or drag a point map directly from the Catalog into this box. You can select a point map with a value domain, or a point map with a ID domain which has a linked attribute table with values. A current limitation of the operation is that Simple Kriging can only handle point maps with a maximum of 89 valid input points; Ordinary Kriging can only handle point maps that contain a maximum of 2500 input points, however within each limiting distance (search radius) only a maximum of 89 valid points are allowed.
Attribute:	In case you selected an input point map with a class or ID domain, select an attribute column (value domain) from the attribute table.
Semi-variogram Model:	Select the model, which should be used to calculate the semi-variogram function $\gamma(h)$. The available models are: Spherical, Exponential, Gaussian, Wave, Rational Quadratic, Circular, and Power. The model/function as well as the variables nugget, sill and range can be found modelling the semi-variogram which is the output of the Spatial correlation operation. For more information, see Spatial correlation : functionality section on semi-variograms, or Graph window : Add Graph Semi-variogram (dialog box).

Nugget: When you found a nugget effect, type a value for the semi-variance γ at distance 0, i.e. the intersection of your semi-variance model and the semi-variance axis (real value ≥ 0).

Sill: Type a value for the sill, i.e. the value where the semi-variance γ approaches the variance of your variable (real value > 0 and sill > nugget).

Range: Type a value for the range, i.e. the distance h at which the semi-variance of your variable is becoming 'stable' (real value > 0).

Slope: When using the Power model, type a value for the linear slope to be used; when a linear model is used, this is the direction coefficient, i.e. $\Delta\gamma/\Delta h$ (real value ≥ 0).

Power: When using the Power model, type a value for the exponent to be used ($0 \leq$ real value ≤ 10). It is advised to use a value between 0 and 2. When you use value 1, the power model will become linear.

Methods: In general, points close to an output pixel will obtain a larger weight value than points farther away.

Simple Kriging: The Simple Kriging method assigns weights to all points of the input map; for the estimation of each output pixel value, all input point values will be used.

Ordinary Kriging: The Ordinary Kriging method uses for the estimation of each output pixel value
only the values of points that are within the user-specified limiting distance towards a pixel. This gives the estimation method a more local character.

Limiting distance: For Ordinary Kriging only: type a value for the limiting distance, also called
search radius or limiting circle. Points that are farther away from any output pixel than the limiting distance are assigned weight zero; the values of these points will thus not be used in the calculation of the output value for that pixel and they will not influence the Kriging equations either. The distance should be positive and normally less than the range of the semi-variogram.

Minimum nr of points:
For Ordinary Kriging only: type a value for the minimum number of points to make sure that the estimation is based at least this many points. When for an output pixel, not enough points are found within the specified limiting distance, then the undefined value will be assigned to the output pixel. It is advised to use at least 4 points.

Maximum nr of points:
For Ordinary Kriging only: type a value for the maximum number of points that should be used by the calculation. When for an output pixel, more points are found within the limiting

	distance than specified, then only the points nearest to the output pixel will be used in the calculation. By specifying a rather small maximum, the algorithm may be faster but the estimation quality may be less.
Remove duplicates:	In case of coinciding points, it is advised to always select this check box. Subsequently, select 'Average' or 'First value'. You can clear this check box when you are sure that you have no coinciding points. If the check box is cleared and any coinciding points, i.e. duplicates, are found, Simple Kriging will fail, while Ordinary Kriging will result in undefined values in the neighbourhood of the coinciding points.
Average:	For points that are less than the specified tolerance distance apart, use the mean value of these points.
First value:	For points that are less than the specified tolerance distance apart, only use the value of the first point encountered.
Tolerance (m):	Type a value (meters) for the tolerance distance: points that are found less than this distance apart are considered coinciding points or duplicates.
Output raster map:	Type a name for the output raster map that will contain the Kriging estimates.
Georeference:	Select the name of an existing georeference or create a new georeference.
Value range:	Accept the default value range, or specify your own range of possible values in the output map. It is advisable to make the value range for the output map wider than the input value range; as negative weights may be used, Kriging estimates may be greater or smaller than your original input values. Mind: in case estimates are calculated that fall outside the specified value range, the pixel will be assigned the undefined value.
Precision:	Accept the default precision of output values, or specify your own precision.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.
Error map:	Select this check box when you also want to obtain an error map. The error map will contain the standard error of the estimates, i.e. the square root of the error variance. Subsequently, type a name for the error map; this name should be different from the name specified for the Kriging output map. Pixel values in the output error map will have value 0

when an input point exactly coincides with the center of the pixel.

A dependent output map is created. Optionally, a second output map containing standard errors can be created.

Command line

The Kriging operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*)
OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*, ErrorMap)
OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*, ErrorMap, `No` | `Average` | `Firstval`)
OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*, ErrorMap, `Average` | `Firstval` , *Tolerance*)
OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*, , `No` | `Average` | `Firstval`)
OUTMAP = `MapKrigingSimple`(InputPointMap, Georef, *SemiVarModel*, , `Average` | `Firstval` , *Tolerance*)

OUTMAP = `MapKrigingOrdinary`(InputPointMap, Georef, *SemiVarModel*, *LimDist*)
OUTMAP = `MapKrigingOrdinary`(InputPointMap, Georef, *SemiVarModel*, *LimDist*, ErrorMap)

OUTMAP = `MapKrigingOrdinary`(InputPointMap, Georef, *SemiVarModel*, *LimDist*, ErrorMap, *min*, *max*)
OUTMAP = `MapKrigingOrdinary`(InputPointMap, Georef, *SemiVarModel*, *LimDist*, ErrorMap, *min*, *max*, `No` | `Average` | `Firstval`)
OUTMAP = `MapKrigingOrdinary`(InputPointMap, Georef, *SemiVarModel*, *LimDist*, ErrorMap, *min*, *max*, `Average` | `Firstval`, *Tolerance*)

where:

OUTMAP	is the name of the output raster map.
`MapKrigingSimple`	is the command to start the Kriging operation, using the Simple Kriging method.
`MapKrigingOrdinary`	is the command to start the Kriging operation, using the Ordinary Kriging method.
InputPointMap	is the name of the input point map with a value domain.
Georef	is the name of an existing georeference for the output raster map.
SemiVarModel	*Model(nugget, sill, range)* | `Power`*(nugget, slope, pow)* This expression defines the semi-variogram model that should be used and the expected parameters.

Model	`Spherical`|`Exponential` |`Gaussian`| `Wave`|`RatQuad`|`Circular`
nugget	value for the nugget, according to your semi-variogram; real value ≥ 0.
sill	value for the sill, according to your semi-variogram; real value > 0 and sill > nugget.
range	value for the range, according to your semi-variogram; real value > 0.
slope	when using the Power model, specify a value for the linear slope to be used; when a linear model is used, this is the direction coefficient, i.e. $\Delta\gamma/\Delta h$; real value ≥ 0.
pow	when using the Power model, specify an exponent; 0 ≤ real value ≤ 10; it is advised to use a value between 0 and 2. When you use value 1, the power model will become linear.
LimDist	For Ordinary Kriging: a value for the limiting distance: points that are farther away from an output pixel than the limiting distance will not be used in the Kriging equations.
ErrorMap	Optional parameter to calculate an error map which will contain the square root of the Kriging error variance values, i.e. standard deviations per pixel. When this parameter is not specified, no error map will be calculated.
min, max	Optional parameters to specify the minimum and maximum number of points that should be taken into account per Kriging estimate/prediction, i.e. for the calculation of any output pixel value. When the minimum is not specified, value 1 will be used. When, for an output pixel, less points than the specified minimum are found within the specified limiting distance, no Kriging is performed: the output pixel will be assigned the undefined value. When the maximum is not specified, value 16 will be used. When, for an output pixel, more points than the specified maximum are found within the specified limiting distance, only the specified maximum number of points that are nearest to the output pixel will be used.
`No`|`Average`|`Firstval`	Choose how to handle possible coinciding points. When no method is specified, Average will be used.
`No`	No removal of duplicates. Mind: when there are coinciding points, the Kriging system may become unstable or unsolvable.
`Average`	Values of duplicates are replaced by the average (arithmetic mean); the coordinates of the first point are taken as new position.

`Firstval`	Values of duplicates (and their coordinates) are replaced by those of the first point.
Tolerance	Optional parameter to specify the distance within which 2 points are considered to coincide. When not specified, a tolerance of 0.1 m will be used.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated.

Example
To perform Ordinary Kriging on input map MPX,
- producing output raster map OUTMAP with georef MYGRF,
- using a Spherical Semi-variogram model with nugget 10 m, sill 70 m and range 0.8 km (found by means of SpatCorr),
- using a limiting distance of 0.2 km,
- producing an error map ERM,
- while each Kriging equation should use at least 5 and at most 16 input points (min, max),
- while considering points that are closer to each other than 10 cm as coinciding (tolerance),
- while for coinciding points only the value of the first point encountered should count,

you can use the following expression:

```
OUTMAP= MapKrigingOrdinary(MpX,MyGrf,
        Spherical(10,70,800),
        200,ErM,5,16,Firstval,0.10)
```

Algorithm

Kriging can be seen as a point interpolation, which requires a point map as input and returns a raster map with estimations and optionally an error map.

The estimations or predictions are calculated as weighted averages of known input point values, similar to the Moving Average operation.

The estimate to be calculated, i.e. an output pixel value $\hat{Z}$, is a linear combination of weight factors (w_i) and known input point values (Z_i):

$$\hat{Z} = \sum (w_i \times Z_i)$$

In case the value of an output pixel would only depend on 3 input points, this would read:

$$\hat{Z} = w_1 \times Z_1 + w_2 \times Z_2 + w_3 \times Z_3$$

Thus, to calculate one output pixel value $\hat{Z}$, first, three weight factors w_1, w_2, w_3 have to be found (one for each input point value Z_1, Z_2, Z_3), then, these weight factors can be multiplied with the corresponding input point values, and summed.

In Moving average, the weight factors are simply determined by the distances of the input points towards an output pixel. In Kriging, however, the weight factors are calculated by finding the semi-variance values for all distances between input points and by finding semi-variance values for all distances between an output pixel and all input points; then a set of simultaneous equations has to be solved.
All semi-variance values are calculated by using a user-specified semi-variogram model (based on the output of the Spatial correlation operation). The weight factors are calculated in such a way that the estimation error in each output pixel is minimized.

The optional error map contains the standard errors of the estimates.

Process Simple Kriging

1. Find the valid input points:
   - input points which coordinates are undefined are ignored,
   - input points which value is undefined are ignored,
   - handle duplicates or coinciding points as specified by the user (no, average, first value).

2. Determine the distances between all valid input points (n) and find the semi-variance value for these distances:
   - for each combination of 2 input points, i.e. a point pair, the distance between the points is determined,
   - for each combination of 2 input points, the distance value is substituted in the user-selected semi-variance model, using the user-specified nugget, sill, and range parameters; this gives a semi-variance value.
   - the semi-variance values are filled out in matrix C (as in Equation 1 below),
   - matrix C is inverted as a preparation for calculations in step 4.

3. *For the first output pixel*, determine the distances of this pixel towards all input points, and find the semi-variance value for these distances:
   - semi-variances are determined using the selected semi-variance model and its parameters as above,
   - the semi-variance values are filled out in vector D (as in Equation 1).

4. Calculate the weight factors (vector w):
   - by multiplying the inverted matrix C (result of step 2) with vector D (result of step 3).

   The obtained weight factors apply to the current output pixel only.

5. Calculate the estimated or predicted values for this output pixel:
   - as the sum of the products of the weight factors and the input point values (Equation 4).

6. Optionally, calculate the error variance and standard error for this output pixel:
   - error variance: by multiplying vector w (result of step 4) with vector D (result of step 3),

- standard error or standard deviation: as the square root of the error variance, according to Equation 5b.

7. Consider the next output pixel and repeat steps 3-7, for all output pixels.

Ordinary Kriging

1. Find the valid input points:
   - input points which coordinates are undefined are ignored,
   - input points which value is undefined are ignored,
   - handle duplicates or coinciding points as specified by the user (no, average, first value).

2. *For the first output pixel*, determine the input points (n) which will make a contribution to the output value depending on the specified limiting distance and minimum and maximum number of points:
   - input points that are farther away from this output pixel than the specified limiting distance are ignored,
   - if the number of points found within the limiting distance is smaller than the specified minimum nr of points, assign the undefined value to this output pixel,
   - use only the specified maximum number of points within the limiting distance, and, in case more points are found within the limiting distance than the specified maximum number of points, use only the points that are nearest to this output pixel.

3. Determine the distances between all input points that will make a contribution to this output pixel (result of step 2), and find the semi-variance value for these distances.
   - for each combination of 2 contributing input points, the distance between the points is determined,
   - for each combination of 2 contributing input points, the distance value is substituted in the user-selected selected semi-variance function, using the user-specified nugget, sill, and range parameters; this gives a semi-variance value.
   - the semi-variance values are filled out in matrix C below (eq. 1).

4. Determine the distances of this output pixel towards all input points, and find the semi-variance value for these distances:
   - semi-variances are determined using the selected semi-variance function or model and its parameters as above,
   - the semi-variance values are filled out in vector D (eq. 1).

5. Calculate the weight factors (vector w):
   - by first inverting matrix C (result of step 3),
   - by solving the set of simultaneous equations.

   The obtained weight factors apply to the current output pixel only.

6. Calculate the estimated or predicted values for this output pixel:

- as the sum of the products of the weight factors and the input point values (Equation 4).

7. Optionally, calculate the error variance and standard error for this output pixel:
   - error variance: by multiplying vector w (result of step 4) with vector D (result of step 3), according to Equation 5a.
   - standard error or standard deviation: as the square root of the error variance, according to Equation 5b.

8. Consider the next output pixel and repeat steps 2-8, until all output pixels are done.

Formulae to calculate weight factors

The Kriging weight factors of n valid input points p_i (i = 1…n) are found by solving the following matrix equation:

$$(C) = (w) \bullet (D) \tag{1}$$

or

$$\begin{pmatrix} 0 & g(h_{12}) & g(h_{13}) & \dots & g(h_{1n}) & 1 \\ g(h_{21}) & 0 & g(h_{23}) & \dots & g(h_{2n}) & 1 \\ g(h_{31}) & g(h_{32}) & 0 & \dots & g(h_{3n}) & 1 \\ \dots & \dots & \dots & \dots & \dots & 1 \\ g(h_{n1}) & g(h_{n2}) & g(h_{n3}) & \dots & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_n \\ l \end{pmatrix} = \begin{pmatrix} g(h_{01}) \\ g(h_{02}) \\ g(h_{03}) \\ \\ g(h_{0n}) \\ 1 \end{pmatrix}$$

This matrix equation can be written as a set of n+1 simultaneous equations:

$$\sum_i (w_i \times g(h_{ik})) + l = g(h_{0i}) \qquad \text{for } k=1,\dots,n \tag{2}$$

$$\sum_i w_i = 1 \tag{3}$$

where:

h_{ik}	is the distance between input point pi and input point p_k
h_{0i}	is the distance between the output pixel and input point p_i
$\gamma(h_{ik})$	is the value of the semi-variogram model for the distance h_{ik}, i.e. the semi-variance value for the distance between input points p_i and input point p_k;
$\gamma(h_{0i})$	is the value of the semi-variogram model for the distance h_{0i} , i.e. the semi-variance value for the distance between the output pixel and input point p_i
w_i	is a weight factor for input point p_i
λ	is a Lagrange multiplier, used to minimize possible estimation error

Matrix C thus contains the semi-variances for all combinations of valid input points that will make a contribution to the output pixel value.
Vector w thus contains the weight factors for all valid input points that will make a contribution to the output pixel value.

Vector D thus contains the semi-variances for an output pixel and all combinations of valid input points.
Equation (3) guarantees unbiasedness of the estimates. The solutions w_i minimize the Kriging error variance σ^2.

Formulae to calculate an estimate or predicted value for an output pixel

$$\hat{Z} = \sum_i (w_i + Z_i) \quad (4)$$

where:

$\hat{Z}$	is the estimate or predicted value for one output pixel to be calculated
w_i	is the weight factor for input point p_i
Z_i	is the value of input point p_I

Formulae to calculate error variance and standard error
The error variance is calculated as:

$$s^2 = \sum_i (w_i \times g(h_{01})) + l \quad (5a)$$

The standard error or standard deviation is the square root of the error variance, thus:

$$s = \sqrt{\left(\sum_i (w_i \times g(h_{01})) + l \right)} \quad (5b)$$

where:

σ^2	is the error variance for the output pixel estimate
σ	is he standard error or the standard deviation of the output pixel estimate
h_{0i}	is the distance between the output pixel and input point p_i
$\gamma(h_{0i})$	is the value of the semi-variogram model for the distance h_{0i} , i.e. the semi-variance value for the distance between the output pixel and input point p_i
w_i	is a weight factor for input point p_i
λ	is a Lagrange multiplier, used to minimize possible estimation error

Notes:

- The contents of matrix C depends on the semi-variogram model selected by the user, its parameters nugget, sill and range, and the geometric distribution of the points within the limiting circle in the input map.
- The contents of vector D is determined by the location of the estimated pixel value with respect to the surrounding input points (inside the limiting circle) and the semi-variogram.
- The estimates are computed as linear combinations of the n point sample values with the weights w_i as coefficients (the w_i are found from equation (1)). Therefore the estimates are called 'linear predictors'.
- Equation (3) guarantees unbiasedness of the estimates. The solutions w_i minimize the Kriging error variance σ^2.
- Equation (5) does not contain the sample attribute information. This means that the error variances solely depend on the spatial distribution of the samples and not on their measurement values (the attribute values).

- In the case of Simple Kriging, it is assumed that all input points contribute in some way to the estimate in each pixel. The Kriging matrix has thus a constant value for all pixels estimated and needs to be inverted only once; however the right hand-side D keeps changing.
- In Ordinary Kriging the number of points used ($n \leq N$) and hence the size of the Kriging matrix (`n+1`) will change from pixel to pixel while calculating the output map(s). Hence it is theoretically possible that for each output pixel a new Kriging system of order `n+1` has to be solved. The algorithm also takes care that for each new set of surrounding input points, this set is sorted according to distance from the estimated pixel in order to enable to select the closest points satisfying the max nr of points condition.

References:

- Clark, I. 1979. Practical geostatistics. Applied Science Publishers, London. 129 pp.
- Cressie, N.A.C. 1993. Statistics for spatial data. Wiley, New York. 900 pp.
- Davis, J. C. 1973. Statistics and data analysis in geology. Wiley, New York. 646 pp.
- Deutsch, C.V., and A.G. Journel. 1992. Geostatistical software library and user's guide. Oxford University Press, New York. 340 pp.
- Isaaks, E. H., and R. M. Srivastava. 1989. An introduction to applied geostatistics. Oxford University Press, New York. 561 pp.
- Journel, A. G. and Ch. J. Huijbregts. 1978. Mining geostatistics. Academic Press, London, 600 pp.
- Krige, D.G. Two-dimensional weighted moving average trend surfaces for ore-valuation, in Proc. Symposium on Mathematical Statistics and Computer Applications in Ore Valuation: Journ. South African Inst. Mining and Metallurgy, Johannesburg, 1966, Mar. 7-8, pp. 13-38.
- Matheron, G.F. Principles of geostatistics: Economic Geology, 1963, vol. 58, pp.1246-1266.
- Meer, F. D. van der. Introduction to geostatistics. ITC lecture notes. 72 pp.
- Olea, R.A. 1991. Geostatistical glossary and multilingual dictionary. Oxford University Press, New York.
- Stein, A. 1998. Spatial statistics for soils and the environment. ITC lecture notes. 47 pp.

7.6 Vector operations

7.6.1 Transform vector map

Functionality / Algorithm

The transform operation is identical for all vector maps: polygon map, segment map and point map.

The Transform map operation transforms the XY-coordinates in a vector map from the map's current coordinate system to another target coordinate system. For polygon maps this concerns the polygon boundaries, for a segment map the segments, and for a point map the point locations. The Transform operation can only be used when a transformation between the coordinate systems is possible.

With the transform operation, you can:
- transform a vector map with a *coordinate system projection* with a certain projection, ellipsoid and/or datum to *another coordinate system projection* with a different projection, ellipsoid and/or datum;
- transform a vector map with a *coordinate system projection* with a certain projection, ellipsoid and/or datum to a *coordinate system latlon* with a certain ellipsoid and/or datum (and vice versa);
- transform a vector map with a *coordinate system latlon* with a certain ellipsoid and/or datum to *another coordinate system latlon* with a different ellipsoid and/or datum;
- transform a vector map with a *coordinate system formula* to the *'related coordinate system*' of this coordinate system formula (and vice versa);
- transform a vector map with a *coordinate system tiepoints* to the *'related coordinate system*' of this coordinate system tiepoints (and vice versa).

Furthermore:
- in a coordinate system without an ellipsoid specification, a sphere is assumed
- when in only one of the coordinate systems a datum is specified, then this datum is assumed for both.

For more information on coordinate system types, see ILWIS objects: coordinate systems.

In general, the operation can be used:
- to integrate data which are obtained from different sources and when these data are in different projections,
- to present data or results in a more appropriate projection.

Preparations for using a coordinate system projection
- When you want to integrate data of different projections, first think of the most suitable projection in which you can do your analysis: when area calculations are involved, you should use an Equal Area projection.
  You can either:
  - transform your maps to one existing coordinate system; select this existing coordinate system as the target coordinate system during Transform operations.
  - first create a new coordinate system, specify projection information, and then transform all your maps to that new coordinate system.
- When you want to present data in other projections, you can generally first create a number of new coordinate systems (using File, Create, or by clicking the create button in the dialog box of a Transform operation) and then transform your map to these new coordinate systems. During the Transform operations,

specify a different output map name for each selected target coordinate system. For presentation purposes, you can also interactively change the coordinate system as used by maps displayed in a map window. For more information, see the tips below.

Input map requirements
A transformation is only possible between:

- a coordinate system projection with a known projection, ellipsoid, and/or datum, and
  - another coordinate system projection with another projection, ellipsoid, and/or datum, or
  - a coordinate system latlon with a known ellipsoid and/or datum, or
- a coordinate system latlon with a known ellipsoid and/or datum, and
  - another coordinate system latlon with a known ellipsoid and/or datum, or
- a coordinate system formula, and
  - the 'related coordinate system' of this coordinate system formula, or
- a coordinate system tiepoints, and
  - the 'related coordinate system' of this coordinate system tiepoints.

In each of these combinations, one coordinate system is the current coordinate system of the vector map, and the other is the selected target coordinate system.

Domain and coordinate system of output map
The output vector map uses the same domain as the input vector map.
The output vector map uses the selected target coordinate system; the coordinate boundaries of the output map will be the transformed coordinate boundaries of the input map.

☞ To see the effect of using different projections, it is advisable to display the input map in a map window and the output maps in other map windows; then add a graticule to the map windows; open the Layers menu and select the Add Annotation, Graticule command.

☞ The Transform operations permanently change the projection of your map(s), i.e. for analysis and calculation purposes.
To temporarily view map(s) which are displayed in a map window in another projection, i.e. for presentation purposes, you can:

- create a coordinate system in which you already specify some projection information,
- display your map(s) in a map window,
- drag your new coordinate system to the map window, or choose the Coordinate System command from the Options menu in the map window and subsequently select another coordinate system.

The contents of the map window will be displayed in the new projection. Projection information of the coordinate system as currently used by the map window can be refined by choosing the Coordinate System command from the Edit menu in the map window. Thus, you do not need to use a Transform operation.

☞ Polygon Map: During a transformation, straight lines which are simply defined by a start node and an end node will always remain straight as only the coordinates of the

begin and end node change. Polygon boundaries however generally consist of a start node and an end node and many intermediate points in between. Then, before using a Transform operation, it is strongly advised to increase the number of intermediate points in all polygon boundaries; in this way, the shape of polygons will be preserved during the transformation.

- First extract segments from your polygons (in the Polygon editor or with the Polygons to segments operation),
- Then, use the Densify segment coordinates operation.

After this, use the Transform segment map operation and repolygonize afterwards, or first repolygonize your segments and then use the Transform polygon map operation.

☞ Segment Map: During a transformation, straight lines which are simply defined by a start node and an end node will always remain straight as only the coordinates of the begin and end node change. Segments however generally consist of a start node and an end node and many intermediate points in between. Then, before using a Transform operation, it is strongly advised to increase the number of intermediate points in all segments with the Densify segment coordinates operation. In this way, the shape of segments will be preserved during the transformation.

Algorithm

The XY-coordinate pairs of the polygon boundaries of the input map are copied, and then transformed.

- When both the input coordinate system and the target coordinate system have the same ellipsoid and datum information, then the transformations are calculated via geographic coordinates; i.e. from XY (input) to φ, λ (LatLon) to XY (target).
- When in the input coordinate system and the target coordinate system, ellipsoid and datum information is different, then Molodensky formulas are used to calculate the ellipsoidal transformations and datum shifts, i.e. from XY (input) to φ, λ (input) to φ, λ (target) to XY (target).

Finally, new polygon areas and perimeters are calculated.

Dialog box

Dialog box options

Input vector map:	Select an input vector map. Open the list box and select the desired input map, or drag a polygon map directly from the Catalog into this box. The coordinate system of the input vector map is the coordinate system you will transform coordinate pairs *from*.
Coordinate system:	Select an existing target coordinate system or create a new coordinate system by using the create button. This is the coordinate system you will transform the coordinate pairs of your input map *to*.
Output vector map:	Type a name for the output vector map that will contain the transformed coordinates.

Show:	Select this box if you want the output map to be displayed in a map window when the operation has finished. Clear this box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Transform vector map operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTMAP = `PolygonMapTransform` (InputPolygonMap, CoordinateSystem)
OUTMAP = `SegmentMapTransform` (InputSegmentMap, CoordinateSystem)
OUTMAP = `PointMapTransform` (InputPointMap, CoordinateSystem)

where:

OUTMAP	is the name of your output polygon map.
`PolygonMapTransform`	is the command to start the Transform Polygon Map operation.
`SegmentMapTransform`	is the command to start the Transform Segment Map operation.
`PointMapTransform`	is the command to start the Transform Point Map operation.
InputPolygonMap | InputSegmentMap | InputPointMap	is the name of the input vector map. The coordinate system of the input vector map is the coordinate system *from* which you will transform.
CoordinateSystem	is the name of your target coordinate system; this is the coordinate system *to* which you will transform.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

7.6.2 Transform coordinates

The Transform coordinates dialog box allows you to type XY-coordinates or LatLon coordinates, using a certain input coordinate system; the operation will then show the resulting XY-coordinates or LatLon coordinates for another target coordinate system. The Transform Coordinates dialog box can only be used when a transformation between the coordinate systems is possible.

A coordinate transformation can be useful to check whether a transformation is correct. When typing an XY-coordinate of a point of interest to be transformed, you are able to interactively view the output coordinates for the selected point.

This dialog box appears:
- when you choose Points, Transform Coordinates from the Operations, Vector Operations menu in the Main window, or
- when you double-click the Transform Coordinates item in the Operation-list, or
- when you click a coordinate system in the Catalog with the right mouse button and choose Vector Operations, Transform Coordinates from the context-sensitive menu.

The Transform coordinates dialog box can be used to:
- transform coordinates from a *coordinate system projection* with a certain projection, ellipsoid and/or datum to *another coordinate system projection* with a different projection, ellipsoid and/or datum;
- transform coordinates from a *coordinate system projection* with a certain projection, ellipsoid and/or datum to a *coordinate system latlon* with a certain ellipsoid and/or datum (and vice versa);
- transform coordinates from a *coordinate system latlon* with a certain ellipsoid and/or datum to *another coordinate system latlon* with a different ellipsoid and/or datum;
- transform coordinates from a *coordinate system formula* to the *'related coordinate system*' of this coordinate system formula (and vice versa);
- transform coordinates from a *coordinate system tiepoints* to the *'related coordinate system*' of this coordinate system tiepoints (and vice versa).

Furthermore:
- when in a coordinate system an ellipsoid is not specified, then a sphere is assumed, and
- when in either of the 2 coordinate systems a datum is not specified, then the same datum is assumed.

For more information on coordinate system types, see ILWIS objects : coordinate systems.

Dialog box options

Input coordinate system:	Select an input coordinate system. Open the list box and select the desired coordinate system, or drag a coordinate system directly from the Catalog into this box. If a description exists for this coordinate system, this description will appear just below this Input coordinate system list box.
Input coordinate:	Type the X and Y coordinates of a point of interest.
Output coordinate system:	Select a target coordinate system. Open the list box and select the desired coordinate system, or drag a coordinate system directly from the Catalog into this box. If a description exists for this coordinate

	system, this description will appear just below this Output coordinate system list box.
Output coordinate:	The calculated coordinates for the point of interest in the output projection will be displayed.

Note: If a *coordinate system latlon* is used, coordinates are presented as geographic coordinates.

☞ When you wish to compare coordinate transformation results of *multiple* coordinate systems, you can add the different coordinate systems to the pixel information window. Open a map with 'the input coordinate system' and move the mouse pointer or click at a position of interest; when a transformation from the current coordinate system to the other coordinate system(s) is possible, the pixel information window will show the transformation results.

7.7 Rasterize

7.7.1 Polygons to raster

Functionality

The Polygons to raster operation rasterizes a polygon map. The output raster map always uses the same domain as the input polygon map. This means that the class names, IDs, or values used in the polygon map are also used in the raster map.

For the output raster map, an existing georeference has to be selected or a new one can be created. The georeference determines the number of lines and columns of the output map and the pixel size of the map, see also the examples in the Additional information below. It is strongly advised that vector maps of the same area are rasterized on the same georeference: any map calculation or spatial operation performed later on a combination of raster maps will only make sense if the pixels in these maps refer to the same area on the ground.

When a polygon map has an attribute table or when the domain of the polygon map has an attribute table, the Polygons to raster operation automatically links this attribute table to the output raster map.

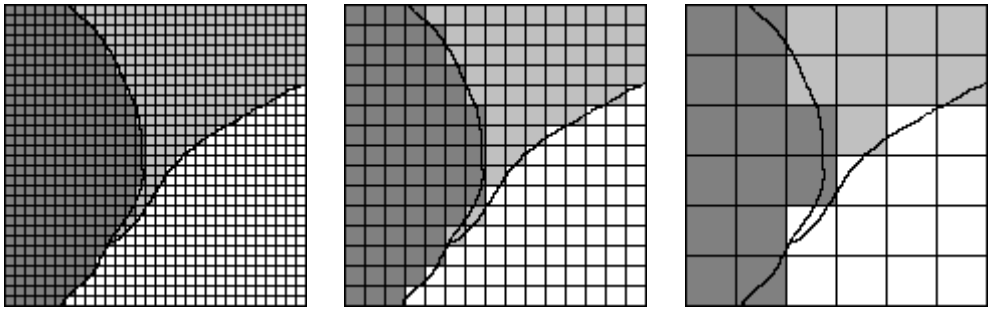
Input map requirements
No special input map requirements. However, at the moment it is not yet possible to rasterize a polygon map with a value domain into a raster map which is stored using 8 bytes per pixel. In that case it is advised to set the value range of the output raster map in such a way that the raster map can be stored with 1, 2, or 4 bytes per pixel.

Domain and georeference of output map
The output raster map uses the same domain as the input polygon map.
The georeference for the output map has to be selected or created; you can usually select an existing georeference corners. The georeference for the raster map must

use the same coordinate system as the polygon map. Georeference None cannot be selected for the output map.

☞ To obtain frequency information on polygons, you can calculate the histogram of a polygon map with the Histogram operation.

Additional information
The georeference you use for the output raster map determines the pixel size of the raster map and thus whether shapes of polygons are well retained. Below you find three examples of rasterized polygon maps, each one with a different pixel size.



Dialog box

Dialog box options

Input polygon map:	Select an input polygon map. Open the list box and select the desired input map, or drag a polygon map directly from the Catalog into this box.
Output raster map:	Type a name for the output raster map that will contain the rasterized polygons.
Georeference:	Select an existing georeference for the output raster map; open the list box by clicking it. Or create a new georeference by clicking the little create button. You can usually select an existing georeference corners.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Polygons to raster operation can be directly executed by typing the following expression on the command line of the Main window:

OUTMAP = `MapRasterizePolygon`(InputPolygonMapName, Georeference)

where:

OUTMAP	is the name of your output raster map.
`MapRasterizePolygon`	is the command to start the Polygons to raster operation.
InputPolygonMapName	is the name of your input polygon map.
Georeference	is the name of an existing georeference. At the moment, it is not yet possible to create a new georeference on the command line.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Polygons to raster operation rasterizes a polygon map. The output raster map always uses the same domain as the input segment map. The user has to specify the georeference for the output raster map.

Process

For each polygon, the corresponding pixels in the raster map are found; these pixels are assigned the class name, ID or value of the polygon.
Other pixels obtain the undefined value.

- The pixel size, indicated in the georeference for the output map, has a large influence on the size (in bytes) of the output raster map. Mind that rasterising on a pixel size of 10 m instead of 50 m. increases the size of the raster map by a factor 25.
- If you want to rasterize only a portion of your polygon map, you can select or create a georeference that does not cover the total area of your polygon map during rasterisation.

For ILWIS 1.4 users

- During rasterisation, an .INF table is not produced anymore. To know the area of polygons, length of segments, nr. of points etc., you can calculate the histogram of a polygon, segment or point map with the Histogram operation.

7.7.2 Segments to raster

Functionality

The Segments to raster operation rasterizes a segment map. The output raster map always uses the same domain as the input segment map. This means that the class names, IDs, or values used in the segment map are also used in the raster map.

For the output raster map, an existing georeference has to be selected or a new one can be created. The georeference determines the number of lines and columns of the output map and the pixel size of the map, see also the examples in the Additional information below. It is strongly advised that vector maps of the same area are rasterized on the same georeference: any map calculation or spatial operation performed later on a combination of raster maps will only make sense if the pixels in these maps refer to the same area on the ground.

When a segment map has an attribute table or when the domain of the segment map has an attribute table, the Segments to raster operation automatically links this attribute table to the output raster map.

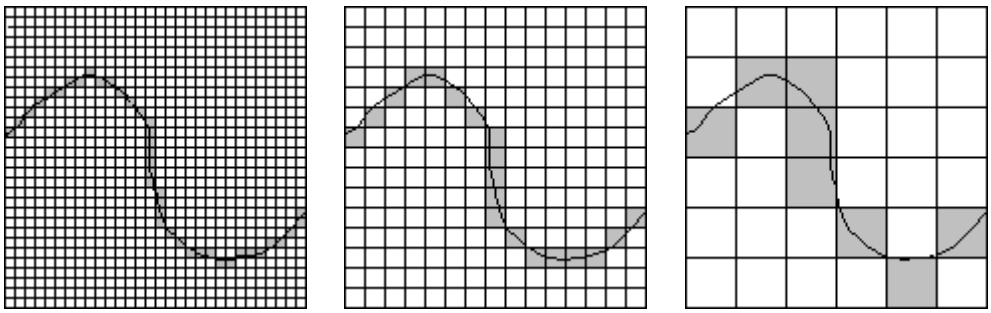
☞ When you want to create a Digital Elevation Model, you can directly do the operation Contour interpolation. Contour Interpolation first rasterizes your segment map, then calculates interpolated values for the pixels in between the contour lines.

Input map requirements
No special input map requirements.

Domain and georeference of output map
The output raster map uses the same domain as the input segment map.
The georeference for the output map has to be selected or created; you can usually select an existing georeference corners. The georeference for the raster map must use the same coordinate system as the segment map. Georeference None cannot be selected for the output map.

☞ To rasterize only segments with a specific code, perform the Mask segments operation first
☞ When you want to print vector maps for annotation purposes on top of raster maps, print quality will improve if you use the vector maps as they are, thus without rasterising them.
☞ To obtain frequency information on segments, you can calculate the histogram of segment map with the Histogram operation.

Additional information
The georeference you use for the output raster map determines the pixel size of the raster map and thus whether shapes of segments are well retained. Below you find three examples of rasterized segment maps, each one with a different pixel size.



Dialog box

Dialog box options

Input segment map:	Select an input segment map. Open the list box and select the desired input map, or drag a segment map directly from the Catalog into this box.
Output raster map:	Type a name for the output raster map that will contain the rasterized segments.
Georeference:	Select an existing georeference for the output raster map; open the list box by clicking it. Or create a new georeference by clicking the little create button. You can usually select an existing georeference corners.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

☞ If you want to create a Digital Elevation Model, you can directly do the operation Contour Interpolation. Contour Interpolation first rasterizes your segment map, then calculates interpolated values for the pixels in between the contour lines.

Command line

The Segments to raster operation can be directly executed by typing the following expression on the command line of the Main window:

OUTMAP = `MapRasterizeSegment`(InputSegmentMapName, Georeference)

where:

OUTMAP	is the name of your output raster map.

`MapRasterizeSegment`	is the command to start the Segments to raster operation.
InputSegmentMapName	is the name of your input segment map.
Georeference	is the name of an existing georeference. At the moment, it is not yet possible to create a new georeference on the command line.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Segments to raster operation rasterizes a segment map. The output raster map always uses the same domain as the input segment map. The user has to specify the georeference for the output raster map.

Process
For each segment, the corresponding pixels in the raster map are found; these pixels are assigned the class name, ID or value of the segment.
Other pixels obtain the undefined value.

- The specified pixel size in the georeference for the output map, has a large influence on the size (in bytes) of the output raster map. Mind that rasterising on a pixel size of 10 m instead of 50 m. increases the size of the raster map by a factor 25.
- If you want to rasterize only a portion of your segment map, you can either select or create a georeference that does not cover the total area of your segment map during rasterisation, or you can first create a SubMap of the segment map and then rasterize this segment submap.

For ILWIS 1.4 users

- During rasterisation, an .INF table is not produced anymore. To know the area of polygons, length of segments, nr. of points etc., you can calculate the histogram of a polygon, segment or point map with the Histogram operation.

7.8 Vectorize

7.8.1 Raster to polygons

Functionality

The Raster to Polygons operation extracts polygons from units in a raster map. The output polygon map uses the same domain as the input raster map, i.e. the class names or IDs in the input raster map will also be used for the polygons in the output polygon map. No polygons are created for pixels with the undefined value.

The polygons in the output map are derived from areas of pixels in the input raster map:
- where pixels have the same class name or ID or
- where pixels have exactly the same value (only possible via the command line).

You can choose to create polygons:
- from 4-connected pixels: areas of pixels are found where pixels with the same class name, ID or value are horizontally or vertically connected; or
- from 8-connected pixels: areas of pixels are found where pixels with the same class name, ID or value are horizontally, vertically or diagonally connected.

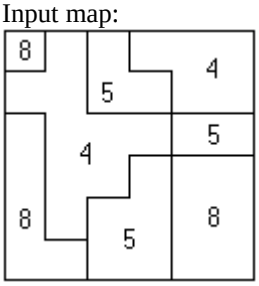
For more information on constructing areas of 4 or 8-connected pixels, see Area numbering : functionality.

Furthermore, you can specify whether or not to smooth the boundaries of the output polygons.

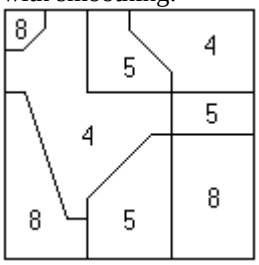
The Raster to Polygons operation attempts to create polygons for neighbouring pixels, which have the same class name or ID or exactly the same value. In an input map which uses a class or ID domain, the areas with a certain class name or ID are clearly distinct, thus, it is rather simple to find the areas. In the dialog box of this operation, you can therefore only select class or ID maps. On the command line, you can also use value maps or images as input. However, in a value map or an image, there are usually no distinct areas with exactly the same value; the values of neighbouring pixels may be similar but are usually not exactly the same. As the operation attempts to find areas where neighbouring pixels have exactly the same value, the resulting output map will usually contain very many areas; these areas may consist of individual pixels or of small groups of a few pixels. To make polygons of a value raster map or an image, it is advised to first use the Slicing operation; then, you can use the output map of the Slicing operation in the Raster to Polygons operation.

Examples
The effect of using 4 or 8-connected pixels with smoothing is illustrated in the figures below.

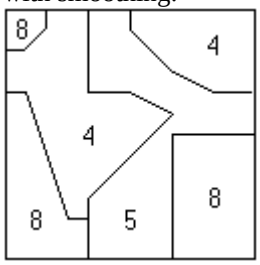
Input map:



Output map 4-connected with smoothing:



Output map 8-connected with smoothing:



The result of this operation depends on the homogeneity of the raster map and the pixel size compared to the size of the mapping units. It is advised to only use this operation on rather homogeneous raster maps that consist of areas with a considerable number of pixels. When you want to extract for instance polygons from a raster map, which is the result of the Classify operation, it may be better to first run the majority filter on that raster map to homogenize the classification results and then perform the Raster to polygons operation.

When the input raster map has an attribute table or when the domain of the input raster map has an attribute table, the Raster to polygons operation automatically links this attribute table to the output polygon map.

Input map requirements
When you use the Raster to Polygons operation through the dialog box, you can use for the input raster map a map with a class, ID or Bool domain. On the command line, you can use any type of input map.
The input raster map must have a georeference, which is not georeference None.

Domain and coordinate system of output map
The output polygon map uses the same domain as the input raster map.
The output polygon map uses the same coordinate system as the georeference of the input raster map. The coordinate boundaries for the polygon map are the boundaries of this georeference.

Dialog box

Dialog box options

Input raster map:	Select an input raster map (map with a class, ID, Bool or Group domain). Open the list box and select the desired input map, or drag a raster map directly from the Catalog into this box. The raster map must have a georeference, which is not georeference None.
Connect:	Specify to obtain polygons from 4-connected pixels or from 8-connected pixels.
4-connected:	Find areas of pixels with the same value/name, which are *horizontally or vertically connected*, then create polygons

	from these areas.
8-connected:	Find areas of pixels with the same value/name, which are *horizontally, vertically or diagonally connected*, then create polygons from these areas.
Smooth lines:	Select this check box if you want the boundaries of the output polygons to be smoothed. Clear this check box if the polygon boundaries should follow the exact boundaries of the pixel areas.
Output polygon map:	Type a name for the output polygon map that will contain the polygons extracted from units in the raster map.
Show:	Select this box if you want the output map to be displayed in a map window when the operation has finished. Clear this box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

7.8.2 Polygons to points

Functionality

The Polygons to points operation creates a point for each polygon in the polygon map. Each point obtains the class name, ID, or value of the corresponding polygon. In this way, polygon label points are created. Optionally, you can also obtain label points for polygons without class names, ID's or values, i.e. for undefined polygons.

☞ By creating polygon label points which also contains label points for undefined polygons, you can easily find the polygons which do not yet have a correct class name, ID or value. To check whether there are any undefined polygons, you can open the output point map as a table.

☞ When you want to edit label points of undefined polygons:

- Display the polygon map and in the Display Options dialog box of the polygon map, choose Boundaries Only;
- Display the point map which contains the polygon labels in the same map window (points for undefined polygons do not appear yet);
- In the map window, open the Edit menu, choose Properties and select the point map. In the appearing Point Map Properties dialog box, click the Break Dependency Link button;
- In the map window, open the Edit menu, choose Edit Layer and select the point map;
- The point editor will be started: all points and their class names, IDs or values will be shown including the label points for undefined polygons;

- Now, you can edit the points with the undefined value to a correct class name, ID or value (depending on the domain of the map). All changes are directly stored.
- When finished, you can repolygonize the segment map using the updated label point file; the class names, IDs or values found in the label point file will be used to assign class names, IDs or values to polygons.

☞ To obtain the names, IDs or values of polygons as text within polygons, it is advised to create an annotation text object and base it on your polygon map. You can edit font, color, and position of the texts in the Annotation Text editor. For more information, see ILWIS objects: annotation text.

Input map requirements
No special input map requirements.

Domain and coordinate system of output map
The output point map uses the same domain as the input polygon map.
The output point map uses the same coordinate system and coordinate boundaries as the input polygon map.

Dialog box

Dialog box options

Input polygon map:	Select an input polygon map. Open the list box and select the desired input map, or drag a polygon map directly from the Catalog into this box. No special input map requirements.
Label points:	For each polygon, a point is created in the output map. The points are located inside the corresponding polygons.
Include undefineds:	Select this check box when the label point file should also contain points for polygons without a class name, ID or value, i.e. for undefined polygons. Clear this check box when you only want to obtain label points for polygons that have a class name, ID or value.
Output point map:	Type a name for the output point map that will contain the polygon label points.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Polygons to points operation can be directly executed by typing the following expression on the command line of the Main window:

OUTMAP = `PointMapPolLabels`(InputPolygonMap)
OUTMAP = `PointMapPolLabels`(InputPolygonMap, `AlsoUndefs`)

where:

OUTMAP	is the name of your output point map.
`PointMapPolLabels`	is the command to start the Polygons to points operation.
InputPolygonMap	is the name of the input polygon map.
`AlsoUndefs`	is an optional parameter to obtain label points for undefined polygons as well.

When the definition symbol `=` is used, a dependent output map is created; when the assignment symbol `:=` is used, the dependency link is immediately broken after the output map has been calculated.

Algorithm

The Polygons to points operation creates a point for each polygon in the polygon map. Each point obtains the class name, ID, or value of the corresponding polygon. In this way, polygon label points are created. Optionally, you can also obtain label points for polygons without class names, ID's or values, i.e. for undefined polygons.

Label points
For each polygon:

- The Y-coordinate of a label point is determined as the mean value of the minimum and maximum Y-coordinate used by the polygon. In other words, the Y-coordinate of the label point will be positioned on the horizontal middle line of the polygon.
- In case the polygon is convex, the X-coordinate of a label point is determined as the mean of the minimum and maximum X-coordinate used by the polygon.
- In case the polygon is concave, it is possible that the horizontal middle line is cut into several pieces by the polygon boundary. Then, the X-coordinate of the label point is positioned at the middle of the longest piece of the horizontal middle line inside the polygon.

This means that:

- for circular polygons, the label point lies more or less at the center of the polygon;
- for vertical half-moon shaped polygons, the label point lies in the middle of the broader part of the polygon.
- for horizontal half-moon shaped polygons, the label point lies in one of the tips of the half-moon.

- for hour-glass shaped polygons, the label point may lie at the narrowest part of the polygon.
- for polygons containing one or more islands, the label point may lie at a narrow part close to an island.

When you are not satisfied with the position of a label point, display the polygon boundaries and the label points in the same map window, break the dependency link of the point map, and use the Point editor to move a label point to another position.

7.9 Table operations

7.9.1 Change domain of table

Functionality

The Change domain of table operation copies the contents of an input table to a new table; the new table will have another domain than the input table.

For the domain of the output table, you can choose:
- domain `None`;
- an existing class or ID domain on disk;
- a class or ID domain of a column in the input table when that column contains unique classes or IDs;
- a class or ID domain of a column in the input table where the column does not contain unique classes or IDs and other column values need to be aggregated (`average`, `minimum`, `maximum`, `sum` and `last` value encountered).

Explanation

1. When you choose to obtain an output table with domain `None`:
   - All records of the input table will be written into the output table.
   - The order of records in the input and in the output table is the same.

2. When you choose to obtain an output table with an existing class or ID domain on disk:
   - The domain items of the input table will be compared with the domain items in the selected domain; when an item (i.e. a class name or an ID) exists in both domains, then the record is written into the output table.
   - The order of records in the output table will be the order of the selected domain.
   - You may use this option: (1) when you have a table containing records for all classes or IDs in a domain, or (2) when you want to obtain a table with only a few records on a few classes or IDs of the input table. For case (2): to create a new domain with only a few classes or IDs (a subset), use the Change domain of table operation and select the domain with the subset.
   - You may also use this option after importing tables and when you know that the imported tables should use the same domain.

3. When you choose to obtain an output table which will use the domain of a column in the table and when each field of that column contains another class name or ID, i.e. when the fields in the selected column are unique:
   - For each unique class name or ID, the values found in other columns in the input table will be written into the output table.
   - The order of records in the output table will be the order of the domain of the selected column.
   - When the class names or IDs in the selected column are not unique and when you do not choose the Aggregate option, an error message will follow.

4. When you choose to obtain an output table which will use the domain of a column in the input table and when the fields of that column do not contain unique class names or IDs, choose the Aggregate option.
   - For each group of class names or IDs found in the selected column, the values found in all value columns will be aggregated by taking either the average, minimum value, maximum value, or the sum; the answers are then written into the output table. You can also choose to simply use the last value encountered per group of class names or IDs.
   - When an aggregation on an input column is not possible, i.e. when an input column does not have a value domain, the last class name, ID, color, etc. encountered per group will be written into the output table.
   - When you use the Average aggregation function, the precision of output value columns will differ from the precision of the input value columns.
   - When you use the Sum aggregation function, the value range of output value columns will differ from the value ranges of the input value columns.
   - The order of records in the output table will be the order of the domain of the selected column.
   - An extra column `Count` will appear in the output table; it contains the number of times that a certain class name or ID was found in the selected column.

Requirements for the input table
No special requirements.

Domain of output table
The domain of the output table will be the domain, which you selected: domain `None`, an existing class or ID domain on disk, or a class or ID domain of a column in the input table.
The output table will contain an extra column Count when an aggregation function is used.

☞ The Change domain of table operation can only use *one* type of aggregation function, which will be applied on *all value columns* in the input table. When you want to use different aggregation functions for various columns or when you want to use weight functions for an aggregation, it is advised to use the Join Column command on the Columns menu of the 'output' table.

Dialog box

Dialog box options

Input table:	Select an input table. Open the list box and select the desired input map, or drag a raster map directly from the Catalog into this box.
Change domain to:	Choose whether you want to change the domain of the table to a class or ID domain of a column in the table or to an existing class or ID domain on disk.
Column:	Select a class or ID column from the table; the domain of this column will be the domain of the output table.
Domain:	Select an existing class or ID domain from disk.
Aggregate:	Select this checkbox when you selected a class or ID column from the table and when the class names or IDs in that column occur more than once, i.e. are not unique. Subsequently, select an aggregation function: `average`, `minimum`, `maximum`, `sum` or `last`. The values of value columns in the input table will appear aggregated in the output table according to the selected aggregation function. Clear this check box when the classes or IDs in the selected column are unique.
Output table:	Type a name for the output table.
Show:	Select this check box if you want the output table to be displayed in a table window when the operation has finished. Clear this check box if you do not want to see this table immediately: you simply define how the output table should be created.
Description:	Optionally, type a description for the output table. The description appears in the title bar when the output table is displayed.

A dependent output table is created.

Command line

The Change domain of table operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTTABLE = `TableChangeDomain`(InputTableName, `None`)
OUTTABLE = `TableChangeDomain`(InputTableName, DomainName)
OUTTABLE = `TableChangeDomain`(InputTableName, ColumnName)
OUTTABLE = `TableChangeDomain`(InputTableName, ColumnName, `avg` | `min` | `max` | `sum` | `last` | `no`)

where:

OUTTABLE	is the name of your output table.
`TableChangeDomain`	is the command to start the Change domain of Table

	operation.
InputTableName	is the name of your input table.
`None`	is a parameter to obtain an output table with domain None.
DomainName	is a parameter to specify an existing class or ID domain on disk; the output table will use this class or ID domain.
ColumnName	is a parameter to specify a class or ID column in the input table; the output table will use this class or ID domain. When you do not perform an aggregation, the specified column must contain unique classes or IDs.
`avg | min | max | sum | last | no`	when the specified column does not contain unique classes or IDs, you can use these parameters to aggregate values of value columns in the table: average, minimum value, maximum value, sum, last value or no aggregation respectively. For non-value columns in the input table, the last class name, ID, or color, etc. encountered will be used.

When the definition symbol = is used, a dependent output table is created; when the assignment symbol := is used, the dependency link is immediately broken after the output table has been calculated.

7.9.2 Table to point map

Functionality

The table to point map operation creates a point map out of a table. The table should have at least two columns, which define the X- and Y-coordinates of the points.

You can choose between the following possibilities:

- the output point map should use the same domain as the table (ID domain); the table will be linked as attribute table to the output point map;
- the output point map will use the domain of a column in the table; the output point map will have no attribute table;
- the output point map should use a new ID domain which is based on the record numbers of the table (domain None) and a user-defined prefix; the table also obtains this new ID domain and the table will be linked as attribute table to the output point map. The new ID domain will automatically obtain the same name of the output point map.

This operation is designed to obtain a point map from data from other packages and which was imported into ILWIS as a table.

Requirements for input table
The input table should have two columns, which contain the X- and Y-coordinates for the points.

Domain and coordinate system of output point map
You can choose whether the output point map should use the ID domain of an ID table, the domain of a column in the table, or whether a new ID domain should be constructed based on the record numbers in a table (domain None) and a user-specified prefix.

Dialog box

Dialog box options

Input table:	Select an input table. Open the list box and select the desired input map, or drag a raster map directly from the Catalog into this box.
X column:	Select a column from the table, which contains the X-coordinates of the points.
Y column:	Select a column from the table, which contains the Y-coordinates of the points.
Coordinate system:	Select an existing coordinate system for the output point map or create a new coordinate system (create button) in which the coordinates of the coordinate columns in the table fit.
Domain of output map:	Choose whether the point map should use the domain of the input table (ID domain), the domain of a column in the table, or
Use table domain:	For a table with an ID domain only, select this option when the output point map should use the ID domain of the table. The table will be linked as attribute table to the output point map.
Use record numbers as IDs:	For a table with domain None only, select this option when a new ID domain should be constructed from the user-specified Prefix and the record numbers of the table. The new ID domain will be used by the output point map; furthermore, the domain of the table will change from domain None to this new ID domain and the table will be linked as attribute table to the output point map.
Use attribute column:	Select this option when the output point map should use the domain of a column in the table. You can select a class, ID, or a value column.
Use column of table:	For a table with domain None only, select this option when the output point map should use the domain of a column in the table. You can select a class, ID, or a value column.
Domain prefix:	When using an input table with domain None and when you selected the option Use record numbers as IDs, type the prefix that should be used for the identifiers in a new ID domain. The new ID domain will contain identifiers

	with this prefix followed by the individual record numbers in the table.
Column:	Select a column from the table; the domain of this column will be the domain of the output point map.
Output point map:	Type a name for the output point map.
Show:	Select this check box if you want the output map to be displayed in a map window when the operation has finished. Clear this check box if you do not want to see this map immediately: you simply define how the output map should be created.
Description:	Optionally, type a description for the output map. The description appears in the title bar when the output map is displayed.

A dependent output map is created.

Command line

The Table to Point Map operation can be directly executed by typing one of the following expressions on the command line of the Main window:

Obtain an ID point map with linked attribute table
(Create it from a table with an ID domain or domain `None`).

OUTMAP = `PointMapFromTable`(InputTableName, CoordinateSystem)
OUTMAP = `PointMapFromTable`(InputTableName, Xcolumn, Ycolumn, CoordinateSystem)

where:

OUTMAP	is the name of the output point map.
`PointMapFromTable`	is the command to start the Table from Point Map operation.
InputTableName	is the name of your input table which has an ID domain or domain `None`. When the table has an ID domain, the output point map will use this ID domain and the table will be linked as attribute table to the output point map. When the table has domain `None`, a new ID domain will be created which contains identifiers with fixed prefix `Pnt` followed by the record numbers of the table. The output point map will use this new ID domain. The domain of the table will change from domain `None` to the new ID domain, and the table will be linked as attribute table to the output point map. The new ID domain will automatically obtain the same name as the output point map.
Xcolumn	is a parameter to specify the name of the column, which contains X-coordinates for the points. When

	the input table already has a column with name X, then you do not have to use the Xcolumn parameter in the expression.
Ycolumn	is a parameter to specify the name of the column, which contains Y-coordinates for the points. When the input table already has a column with name Y, then you do not have to use the Ycolumn parameter in the expression.
CoordinateSystem	is the parameter to specify an existing coordinate system for output point map.

To obtain a point map with the domain of a column in the table

OUTMAP = `PointMapFromTable`(InputTableName, CoordinateSystem, AttributeColumn)

OUTMAP = `PointMapFromTable`(InputTableName, Xcolumn, Ycolumn, CoordinateSystem, AttributeColumn)

where:

InputTableName	is the name of your input table; the table can have any domain.
AttributeColumn	is the name of an attribute column; the output point map will have the domain of this column. The output point map will not have an attribute table.

Obtain an ID point map with linked attribute table

(Create it from a table with domain `None`)

OUTMAP = `PointMapFromTable`(InputTableName, CoordinateSystem, Prefix)

OUTMAP = `PointMapFromTable`(InputTableName, Xcolumn, Ycolumn, CoordinateSystem, Prefix)

where:

InputTableName	is the name of your input table; the table should have domain `None`.
Prefix	is the parameter to specify the prefix for a new output ID domain which will be used both by the output point map and the table. This ID domain is constructed from the user-specified prefix followed by the record numbers of the table. Furthermore, the domain of the input table will change from domain `None` to the newly constructed ID domain, and the input table will be linked as attribute table to the output point map. The new ID domain will automatically obtain the same name as the output point map.

When the definition symbol = is used, a dependent output map is created; when the assignment symbol := is used, the dependency link is immediately broken after the output map has been calculated.

7.9.3 Glue tables

Functionality / Algorithm

The Glue tables operation allows you to glue or merge two or more tables together. As input tables, you may use:

- tables with domain None,
- tables with class domains,
- tables with ID domains,
- tables with class domains and ID domains.

The Glue tables operation should be regarded as a tool to combine different tables. You can for instance combine or integrate attribute tables of different years. Tables with domain None can also be glued vertically, one below the other.

The operation will automatically determine:

- the domain of the output table,
- the domains of the columns in the output table.

Then, fields in the input tables will be copied to the output table.

In the dialog box, you can select up to 4 input tables. On the command line, you can specify as many input tables as you like.

Process

First, the domain for the output table will be determined; the domain for the output table depends on the domains that are used by the input tables:

- when the domain of all input tables is domain None, the output table will also use domain None,
  - when the option Vertical is used, the number of records in the output table will be the sum of all records in all input tables,
  - when the option Vertical is not used, the output table will have the same the number of records as the largest input table.
- when all input tables have the same Class or the same ID domain, the output table will also use this class or ID domain;
- when the input tables use different class domains or different ID domains, then a new output class or ID domain will be created which contains all class names or all IDs of all input table domains; the new output domain will obtain the same name as the output table;
- when some of the input tables use class domains and other input tables use ID domains, then a new output ID domain will be created which contains, as IDs, all class names and all IDs of all input table domains; the new output domain will obtain the same name as the output table.

Then, the columns to be copied to the output table will be examined.

For input tables with a class or ID domain and for input tables with domain None (no vertical gluing):

- when in 2 or more input tables, columns are found that have the same name and the same domain, then only the column of the first input table is copied to the output table,
- when in 2 or more input tables, columns are found that have the same name but which are using different domains, then all these columns are copied to the output table; in the output table, the column of the first table will keep its original name, while the second column will obtain a 2 behind its name, etc. (e.g. MyColumn and MyColumn2) so that you can identify from which table each column was copied,
- other input columns which only exist in one table will always be copied to the output table.

For input tables with domain None and when you selected the option vertical gluing:

- when in 2 or more input tables, columns are found that have the same name and the same domain, then all values of all these columns are copied into a single output column (i.e. below each other = vertical);
- other columns are glued as above.

When a new Class or ID domain is created for the output table, this domain will obtain the same name as the output table. On the command line, you can also specify a name for the output domain yourself.

Examples

Combining tables with domain None and using option Vertical

First input table			Second input table			Output table		
	Direction	Length		Direction	Length		Direction	Length
1	0	2223	1	5	6993	1	0	2223
2	1	4737	2	6	1123	2	1	4737
3	2	2048	3	7	4273	3	2	2048
4	3	6000	4	8	1827	4	3	6000
5	4	0	5	9	1265	5	4	0
						6	5	6993
						7	6	1123
						8	7	4273
						9	8	1827
						10	9	1265

- In this example, both input tables use domain None. The output table will also use domain None.
- Because option Vertical is used, the output table contains the total number of records of all input tables.
- In this example, the columns Direction and Length occur in both input tables; in both the input tables, these columns have the same name and use the same domain. The output table will therefore contain one column Direction and one column Length.
- The values of the columns are then copied to the output table.

Combining tables with domain Class

First input table		Second input table		Output table		
	Landvalue80		Landvalue90		Landvalue80	Landvalue90
Agriculture	75	Agriculture	100	Agriculture	75	100
Agriculture (irri)	125	Agriculture (irri)	150	Agriculture (irri)	125	150
Bare rock	10	Airport	600	Airport	?	600
Bare soils	10	Bare rock	50	Bare rock	10	50
Forest	25	Bare soils	50	Bare soils	10	50
Grassland	25	Forest	75	Forest	25	75
Lake	?	Grassland	75	Grassland	25	75
Riverbed	?	Lake	?	Lake	?	?
Shrubs	35	Riverbed	?	Riverbed	?	?
Urban center	750	Shrubs	50	Shrubs	35	50
Urban periphery	500	Urban center	1000	Urban center	750	1000
		Urban periphery	750	Urban periphery	500	750

- In this example, the input tables use different domains. Therefore, for the output table, a new domain is created which contains all items of the input domains.
- In this example, columns 'Landvalue80' and 'Landvalue90' have distinct names and occur only in one input table, therefore both columns will be copied to the output table.
- The values of columns Landvalue80 and Landvalue90 are then copied to the output table.
- If each of the input tables would have contained a column 'Landvalue' (using the same domain in both input tables), then the output table would have contained only one column 'Landvalue' containing the values of column 'Landvalue' of the first input table.
- If each of the input tables would have contained a column 'Landvalue' (using the different domains in both input tables), then the output table would have contained two columns, called 'Landvalue' and 'Landvalue2'.

Domain of output table

- When all input tables use domain None, the output table will also use domain None.
- When all input tables use the same class domain or the same ID domain, then the output table will also use this class domain or this ID domain.
- When the input tables use different class domains or different ID domains, then a new class domain or a new ID domain will be created for the output table; the output domain will contain all class names or all IDs of the input domains.
- When the input tables use class domains and ID domains, then a new ID domain will be created for the output table; the output domain will contain all class names and IDs of the input domains as IDs.

When a new domain is created for the output table, the domain will obtain the same name as the output table. On the command line, you can also specify a name for the output domain yourself.

Dialog box

Dialog box options

Number of input tables:	Select the number of tables that you want to glue or merge together (2, 3, or 4). In the dialog box, the number of input tables is limited to four. On the command line, this limitation is not present.
First input table:	Select the first input table. Open the list box and select the desired input table, or drag a table directly from the Catalog into this box.
Second input table:	Select the second input table.
Third input table:	Optionally, select a third input table.
Fourth input table:	Optionally, select a fourth input table.
Output table:	Type a name for the output table that will contain all input tables.

Show:	Select this check box if you want the output table to be displayed in a table window when the operation has finished. Clear this check box if you do not want to see this table immediately: you simply define how the output table should be created.
Description:	Optionally, type a description for the output table. The description appears in the title bar when the output table is displayed.
Vertical:	In case you are merging tables that use domain None: select this check box to create an output table which will contain as many records as the total number of records of all input tables. If a column exists in more that one input table with the same name and the same domain, then the values of these columns will be glued into one output column (values one below the other = vertically). In case you are merging tables that use domain None: clear this check box when the output table should contain the same number of records as the first input table.

A dependent output table is created. When the input tables use different class or ID domains, a new domain will be created for the output table. This output domain will contain all class names and/or IDs of the input domains and the domain will obtain the same name as the output table.

Command line

The Glue tables operation can be directly executed by typing one of the following expressions on the command line of the Main window:

OUTTABLE =	`TableGlue`(FirstInputTableName, SecondInputTableName [, MoreInputTables])
OUTTABLE =	`TableGlue`(FirstInputTableName, SecondInputTableName [, MoreInputTables] , `Vertical`)
OUTTABLE =	`TableGlue`(NewDomain, FirstInputTableName, SecondInputTableName [, MoreInputTables])

where:

OUTTABLE	is the name of your output table.
`TableGlue`	is the command to start the Glue tables operation.
FirstInputTableName	is the name of the first input table.
SecondInputTableName	is the name of the second input table.
MoreInputTables	optionally, you can specify more input table names, delimited by commas.
`Vertical`	in case the input tables use domain `None`, an optional parameter to specify that the input tables should be

NewDomain	in case of merging two or more Class or ID tables that do not have the same domain: an optional parameter to specify a name for the new output domain in which all input domain items will be merged. When the Class or ID domains of input tables are not the same and this parameter is not specified, a new output domain will be automatically created with the same name as the output table.

On the command line, you can specify as many input tables as you like, i.e. you can merge as many tables as you like. When using the dialog box of this operation, only two, three or four input tables can be merged at a time.

When the definition symbol = is used, a dependent output table is created; when the assignment symbol := is used, the dependency link is immediately broken after the output table has been calculated. When tables are merged which use different class or ID domains, a new domain will be created for the output table.