CHAPTER 9

Spatial data analysis: neighbourhood and connectivity calculations

In the previous chapter you have seen a number of basic spatial analysis operations used for the overlay of raster maps. Where the overlay operations only consider the combination of raster cells in different maps at the same location, the *neighbourhood* calculations evaluate the characteristics of a specified location and its *surrounding* area. These calculations make use of a small calculation window (e.g. 3x3 cells) that repeats a specified calculation on every pixel in the map, taking into account the values of its neighbours. When only the 4 direct neighbours are used (i.e. the ones that are on the same line or the same column as the central pixel), we talk about a 4-connected operation. When all 8 neighbours are taken into account, it is an 8-connected operation (see Figure 9.1).

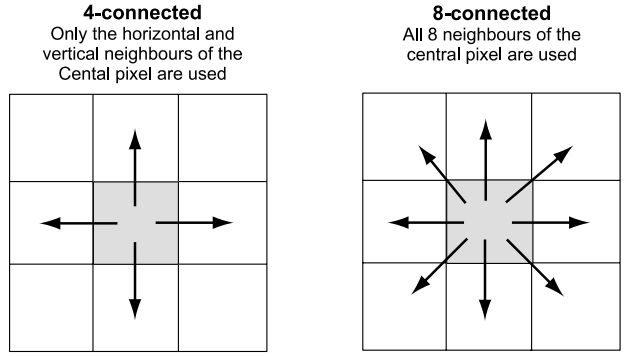


Figure 9.1: The difference between 4-connected and 8-connected neighbourhood calculations.

The calculation window starts with the first pixel on the first row of the map. The result of the calculation is stored in the central pixel. Then the calculation window moves to the second pixel in the first line, and the calculation is repeated. In this way the calculation window moves over the entire map.

For some of the calculations, an iterative procedure is required, in which the calculation window, after finishing the calculation at the last pixel in the last line, will start a reverse operation, using the data of the previous run as input. The calculation will be repeated several times until no more changes occur in the map or a predefined number of iterations.

The following neighbourhood calculations will be explained in this chapter:
- Filtering (in chapter 6 already explained for satellite images) applied to thematic maps. You will look at those filters that are especially used for raster maps, such as smoothing filters, counting filters, etc.;
- Neighbourhood calculations in Map Calculation formulas. The use of powerful neighbourhood functions and map iterations will be shown, that allow you to generate models that evaluate the conditions of neighbouring pixels in a map;
- Distance calculation, in which the calculation window is used to calculate the distance from source pixels in the map (which could be roads, rivers, etc.);
- Area numbering, in which unique identifiers are assigned to groups of connected pixels with the same class name.

Some other neighbourhood calculations will be discussed in other chapters:
- Chapter 10 will discuss the specific calculations that are used for the analysis using Digital Elevation Models, including Contour Interpolation and the use of gradient filters in the calculation of slope length and slope direction.
- Chapter 11 will treat the various Point Interpolation calculations available in ILWIS.

In the last part of this chapter some examples are given of how ILWIS can be used for performing connectivity calculations. By using these calculations, spatial units that are connected (using a set of pre-defined rules) are characterized.

Before you can start with the exercises, you should start up ILWIS and change to the subdirectory C:\ILWIS 3.0 Data\Users Guide\Chapter09, or to the directory where the data files for this chapter are stored.

- Double-click the ILWIS icon on the desktop.
- Use the Navigator to go to the directory: C:\ILWIS 3.0 Data\Users Guide\Chapter09.

9.1 Filtering

In chapter 6, the filter operation has been discussed for the use on satellite images. *Filtering* is a process in which each pixel value in a raster map is replaced with a new value, provided that a condition or a set of conditions is satisfied. The new value is obtained by applying a certain function to each input pixel and its neighbours. There are many types of filters available in ILWIS. Some of these are especially designed for the use on thematic or value maps, some are especially for satellite images, and some may be used for both.

The following filters are useful to apply on maps:

- **Filters used on Digital Elevation Models**. With the help of the filters working on DEMs, you can calculate slope steepness, slope direction, slope convexity/concavity and hillshading. These filters will be treated in the next chapter, which is completely dedicated to the generation and use of DEMs;
- **Majority filters**. These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name, of each input pixel and its 8 neighbours (when using a 3x3 filter);
- **Smoothing filters**. This filter is used on value maps and will calculate for each pixel the average value of the central pixel and its 8 neighbours (when using a 3x3 filter);
- **Rank order filters**. This filter is used on value maps and can be used to calculate the median value or any other rank of the central pixel and its 8 neighbours (when using a 3x3 filter);
- **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 value. These filters are used in what is called *mathematical morphology*. You can use them for *dilation* (the growing of “true” pixels in a certain direction) and *erosion* (the shrinking of “true” pixels in a certain direction);
- **Counting filters**. Counting filters are used to count the number of pixels within a certain search radius around each central pixel. They can be used to count the number of point features, such as houses, wells or landslides.

You can also define filters of other sizes. In the following sections examples will be given of the use of these different filters.

Majority filters

These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name, of each input pixel and its neighbours. You can use this filter to reduce a large variation in classes or values for neighbouring pixels.

A simple example will be used in this exercise: a small suitability map. It has 41 rows and 54 column so that you can clearly see the effect of the filter on the map. The map `Suitability` is a value map, and contains the values 1 to 6 for showing the suitability of an area for growing maize. The area that has not been rated is undefined.

- Display the raster map Suitability with the Pseudo Representation.
- Find out the values of the different pixels by clicking them.
- Close the map window.

Now we will apply a *majority* filter. The standard majority filter MAJORITY is a 3x3 filter. As output value for each central pixel, the filter selects the predominant (most frequently occurring) value or class name, of each input pixel and its 8 neighbours. If all 9 pixels have a different value or class, the first value or class name encountered is used as output.

- Click the map Suitability with the right mouse and select Image Processing, and the command Filter from the context-sensitive menu. The Filtering dialog box is opened.
- Select the Filter Type: Majority, and the Filter Name: Majority.
- Type for the Output Raster Map name: Suitability1 and click the Show button.
- In the Display Options - Raster Map dialog box select Representation Pseudo and click OK. The map is now displayed.
- Compare the maps Suitability1 and Suitability and close the two map windows afterwards.

The result of the majority filter is that single undefined pixels that were within the different areas are now filled up, and the linear features of 1 pixel wide have disappeared.

There is also a majority filter that is used only on undefined values. The *undef-majority* filter (MAJUNDEF) is a 3x3 filter that selects the predominant value or class name if the central pixel is undefined. If the central pixel is defined, the value or class name will not be changed.

- Now filter the map Suitability with the majority filter Majundef, and create output map Suitability2.
  Display all three maps (Suitability, Suitability1, and Suitability2) next to each other and compare them.

The result of the undefined majority filter, is that single undefined pixels disappear and the mapping units expand into the undefined areas.

Similarly to the undefined majority filter, there is a *zero majority* filter (MAJZERO), which only selects the predominant value or class name if the central pixel has value zero. If the central pixel is not zero, the value or class name remains the same.

- Close the three maps (Suitability, Suitability1, and Suitability2).

Smoothing filters

These filters calculate for each pixel the average value of the central pixel and its 8 neighbours (if using a 3x3 filter). To see the effect of smoothing filters, we will change the undefined values of the map Suitability into zero values first, before running the filter AVG3X3. This filter calculates the average value of each 9 pixel values considered (3x3 matrix).

- Type the following formula on the Command line:
  Suitability_null = IFUNDEF(Suitability, 0) ↵
- Click Show in the Raster Map Definition dialog box, and select the Pseudo Representation in the Display Options dialog box. Check whether the undefined values are now zero.
- Filter map Suitability_null with Filter Type Average, with 3 Rows and 3 Columns. Create Output Raster Map Suitability3 and change the Precision to 0.1.
- Display map Suitability3 next to map Suitability_null and compare them.
- Click the pixels in both maps to find out their values.

You can clearly see that the sharp edges of the filtered map Suitability3 are blurred now. Whereas the unfiltered map Suitability_null contains the integer values from 0 to 6, the filtered map has intermediate values. The smoothing will even be larger when we use a larger filter size (say 5x5).

- Filter map Suitability_null with the Average Filter Type, now with 5 Rows and 5 Columns.
- Create Output Raster Map Suitability4 and change the Precision to 0.1.
- Display map Suitability4 with Representation Pseudo next to maps Suitability_null and Suitability3 and compare them. Click the pixels in both maps to find out their values.
- Close all the maps when you are finished.

Rank order filters

These filters can be used to calculate for instance the median value of the central pixel and its 8 neighbours (if using a 3x3 filter).

- Filter map `Suitability_null` with the Filter Type `Rank Order` and Filter Name `Med3x3`. Create Output Raster Map `Suitability5`. Do not change the Precision since that will have no effect with a median filter.
- Display map `Suitability5` next to map `Suitability_null` (both with the `Pseudo Representation`) and compare them. Click the pixels in both maps to find out their values.
- Close all maps when you are finished.

The result of this filter is rather similar to the output of the majority filter. The effect can be better seen when we filter a map with more diverse values, such as the `Slope` map.

- Filter map `Slope` with the `Rank Order` Filter Type and the Filter Name `Med5x5`. Create Output Raster Map `Slope_filter`. Do not change the Precision.
- Display map `Slope_filter` next to map `Slope` (both with the `Pseudo Representation`) and compare them. Click the pixels in both maps to find out their values.
- Close all maps when you are finished.

Binary filters

Binary filters regard the input map as a binary map. This means that zero and undefined values are regarded as zero and all other values as one. Depending on the central pixel value and its 8 neighbours, the filter produces either value zero or value one as output values. These filters are used in what is called *mathematical morphology*. You can use them for *dilation* (the growing of "true" pixels in a certain direction) and *erosion* (the shrinking of "true" pixels in a certain direction). Let us try some of these filters, using the map `Suitability_null`. First we will use the `Dilate8` filter, which will make the suitable areas grow in size.

- Filter map `Suitability_null` with the `Binary` Filter Type, and the Filter Name `Dilate8`. Create Output Raster Map `Suitability6`. Note that the Domain of the Output Raster Map is `Bool`.
- Display map `Suitability6` next to map `Suitability_null` (the last one with the `Pseudo Representation`) and compare them.
- Click the pixels in both maps to find out their values.
- Close the map windows.

The Dilate8 filter assigns a 1 to the central pixel if any of the neighbours is "true". The result of using Dilate8 is that true pixels "grow" in all directions. The reverse will be done using the Shrink8 filter. The shrink filter assigns a 1 only when the central pixel is 1 and all neighbours are also 1. The result of using the Shrink8 filter is that areas of "true" pixels "shrink" along their edges; single lines of "true" pixels and single "true" pixels totally surrounded by "false" pixels disappear. This filter will make the suitable area shrink in size in all directions. If you use the Shrink4 filter the area will shrink in horizontal and vertical direction, but not in a diagonal direction.

- Filter map Suitability_null with the Binary Filter Type and the Filter Name Shrink8. Create Output Raster Map Suitability7. Note that the Domain of the Output Raster Map is Bool.
- Display map Suitability7 next to map Suitability_null and compare them.
- Click the pixels in both maps to find out their values.
- Close the map Suitability7.

Next, we will use some filters that extract boundaries of units in a map. For example, the filter Inbnd8 assigns a 1 when the central pixel is 1 and at least one of the 8 neighbours is 0 (meaning it is located at the edge of a group of "true" pixels). The result of using this filter is that areas of 8-connected true pixels are replaced by their 8-connected outline (an 8-connected boundary of true pixels).

- Filter map Suitability_null with the Binary Filter Type and the Filter Name Inbnd8. Create Output Raster Map Suitability8. Note that the Domain of the Output Raster Map is Bool.
- Display map Suitability8 next to map Suitability_null and compare them. Click the pixels in both maps to find out their values.
- Close the map Suitability8.

The reverse of the Inbnd8 filter is the Outbnd8 filter, which extracts the outer boundary of areas of 8-connected pixels (3x3 matrix). So instead of extracting the boundary of the "true" pixels (pixels with a value larger than 0) it will extract the boundary of the "false" pixels (pixels with a value 0).

- Filter map Suitability_null with the Binary Filter Type, and the Filter Name Outbnd8. Create Output Raster Map Suitability9. Note that the Domain of the Output Raster Map is Bool.
- Display map Suitability9 next to map Suitability_null and compare them. Click the pixels in both maps to find out their values.
- Close all map windows.

The last of the binary filters, which we would like to show you, is called the Conn8to4. When "true" pixels are 8-connected, this filter makes them 4-connected by adding a "true" pixel in some places. This gives your binary image a better outlook. It is also used to improve line barriers used in the distance calculation. The line barrier must be at least two pixels wide, so that no pixels on either side of the line are connected, not even diagonally. This will be demonstrated using the Drainage map.

- Click with the right mouse button on segment map Drainage and select Rasterize, Segment to Raster from the context-sensitive menu. The Rasterize Segment Map dialog box is opened.
- In the Rasterize Segment Map dialog box, select GeoReference Cochabamba and click the Show button
- Click OK in the Display Options dialog box. The map is displayed.
- Zoom in on a part of the map containing drainage lines to see that the lines are sometimes 8-connected. The undefined pixels neighbouring these drainage lines are thus also 8-connected.
- Filter the map Drainage with the Binary Filter Type and the Filter Name Conn8to4. Create Output Raster Map Drain4connected. Note that the Domain of the Output Raster Map is Bool.
- Display map Drain4connected next to map Drainage.
- Zoom in on a small section of both maps with the same drainage lines and compare them. You can see that all drainage lines are now 4-connected. In a distance calculation they can now act as a closed barrier.
- Close all map windows.

User-defined linear filters: Example of a counting filter

Besides the standard filters that are stored in the ILWIS system directory, of which we have seen some in the previous part of this chapter, it is also possible to define filters yourself.

One example of a user-defined filter is a *counting filter*. Counting filters are used to count the number of defined pixels, within a certain search radius of a central pixel. They can be used to count the number of point features, such as houses, wells or landslides. We will create a counting filter that will count the number of landslide pixels within a radius of 10 pixels (Figure 9.2).

The filter shape resembles a circle in which all the pixels within the circle (with a radius of 10 pixels) contain the value 1, and the rest the value 0. When we apply this filter on a binary map, showing the landslides, for each pixel the number of pixels with landslides within the 10 pixel radius is calculated.

rows \ columns	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
-10	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
-9	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
-8	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
-7	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
-6	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
-5	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
-4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
-3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
-2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
5	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
6	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
7	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
8	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
9	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0

Figure 9.2: User-defined counting filter of size 21 * 21.

You will first have to create the binary map with the landslides, using the information from the geomorphologic map `Geomorphology`.

- Type the following formula on the Command line of the Main window:
  `Landslide = IFF(Geomorphology = "AL", 1, 0)` ↵

  The Raster Map Definition dialog box is opened. Click the Show button and click OK in the Display Options – Raster Map dialog box. The map is now displayed.
- Check the map values and close the map.
- Open the `Counting` filter and have a look at it. Note that there is a Gain of `1.0` specified.

The *gain* is the value with which the result of the filter will be multiplied. When a linear filter is placed over the raster map, the filter will multiply all the values of the filter with the corresponding pixel values in the same position in the map. In our case this will result either in a 0 or in a 1. Then all the results of the individual filter cells are summed up and multiplied by the gain. For a counting filter the gain should be 1, since we want to know how many pixels there are with landslides, within the specified search radius of 10 pixels around every pixel. The resulting value will be

stored in the central pixel, then the filter shifts one pixel further and does the same calculation.

- Click the map `Landslide` with the right mouse button and select Image Processing, and the command Filter. The Filtering dialog box is opened.
- In the Filtering dialog box select Filter Type `Linear` and Filter Name `Counting` and type the name of the Output Raster Map: `Landslidenr`.
- Note that the value domain is automatically set with the maximum value being the total number of 1 values in your filter.
- Click Show in the Filtering dialog box. The map `Landslidenr` is now calculated and the Display Options dialog box is opened
- Click OK. The map is displayed.
- Check the values in the map by clicking on the pixels.
- After you have analyzed the result, close the map window.

Summary: Filter operation using maps

The following filters are useful to apply on maps:

- Filters used on Digital Elevation Models. With the help of the filters working on DEM's you can calculate slope steepness maps, slope direction maps, slope convexity/concavity and hillshading maps.

- Majority filters. These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name of each input pixel and its neighbours.

- Smoothing filters. These filters will calculate for each pixel the average value of the central pixel and its neighbours.

- Rank order filters. These filters can be used to calculate the median value of the central pixel and its neighbours.

- Binary filters. These filters are used in what is called *mathematical morphology*. You can use them for *dilation* (the growing of "true" pixels in a certain direction) and *erosion* (the shrinking of "true" pixels in a certain direction).

- Counting filters. Counting filters are used to count the number of pixels within a certain search radius around each central pixel. They can be used to count the number of point features, such as houses, wells or landslides.

9.2 Neighbourhood calculations using Map Calculation

Neighbourhood calculations are a special type of spatial analysis in ILWIS. They are calculations on pixels where the outcome of a calculation depends directly on the values of the neighbouring pixels. Neighbourhood calculations can be performed on user-selected pixels and on whole maps, not like normal filters that can only be applied on a whole map.

Like in a filtering procedure, neighbourhood calculations make use of a calculation window. This imaginative window of 3x3 cells moves over the raster map and each cell of the output map is calculated according to the specified neighbourhood expression.

To be able to perform calculations on user-selected pixels, a pixel neighbour position is coded with respect to the central pixel. The neighbours in this so-called 'matrix' (Figure 9.3) are coded from 1 to 9, where the central pixel has a value of 5.

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Figure 9.3: The 3x3 neighbourhood matrix with the identifiers for the positions of the neighhour pixels (1, 2, 3, 4, 6, 7, 8, 9) with respect to a central pixel (5).

We can distinguish two types of neighbourhood calculations:
- neighbourhood calculations on the value of a *single* neighbour;
- neighbourhood calculations using neighbourhood *functions* which will calculate with the values of *multiple* neighbours.

These two types can also be combined.

9.2.1 Calculating with the value of a single neighbour

The simplest neighbourhood expression retrieves the value of a single user-selected neighbour:

`Outmap = Map#[`*Neighbour*`]`

where:

`Outmap`	is the name of the output map.
`Map#`	is the syntax to specify the name of the input map `Map` over which a neighbourhood matrix will be moved.
`[`*Neighbour*`]`	is the syntax to select *a single* neighbour from the map over which the neighbourhood matrix is moved. *Neighbour* is a number between `1` and `9` which indicates the position of a certain pixel in the neighbourhood matrix as numbered as in Figure 9.3; the value of the selected neighbour will be retrieved. Note that the position of the single neighbour is enclosed in *square* brackets!

- Type the following formula on the Command line of the Main window (note the *square* brackets!):
  `Dem_uppervalue = Dem – Dem#[2]` ↵
- Click Show in the Raster Map Definition dialog box. The map is calculated.

By this calculation, for every pixel, the value of a pixel from map `Dem` will be retrieved (first part of expression), and simultaneously, a neighbourhood matrix will be moved over map `Dem` (second part of expression), which will retrieve the value of the neighbour on position [2]. These two values will then be subtracted from one another. Refer to Figure 9.3 for the positions of neighbours. So, in this calculation the value of a pixel in the first input map is subtracted by the value of the pixel on position [2] (upper center) as found by the matrix which moves over the second map.

- The Display Options – Raster Map dialog box is opened. Click OK.
- Check the result and close the map window.

In the next example you will calculate the altitude difference of a map in the X and Y direction. The `Dx` (difference in X direction) and `Dy` (difference in Y direction) maps are the output maps using a Digital Elevation Model (`Dem`) as input map.

- Type the following formula on the Command line of the Main window:
  `Dx = Dem#[4] - Dem#[6]` ↵

This formula means: Subtract the value of the pixel on the [6] position (the right neighbour of the central pixel) from the value of the pixel on the [4] position (the left neighbour of the central pixel). The result will be assigned to the central pixel in map `Dx`.

- Press Show in the Raster Map Definition dialog box. The map is calculated and the Display Options dialog box is opened. Select Representation `Pseudo` and click OK.
- Check the result with the pixel information window (using the map `Dem` and the map `Dx`).
- Close the pixel information window and the map window.
- Now calculate a map `Dy` that gives the difference in Y direction.

9.2.2 Using neighbourhood functions on multiple neighbours

This section deals with *neighbourhood functions*. A neighbourhood function can calculate with all values found by a neighbourhood matrix, or only with the values of some selected neighbours in the matrix. The syntax for neighbourhood function NBFLT will be treated separately. At the end of this section, also iterative calculations will be explained.

Neighbourhood functions on multiple neighbours

`OutMap = NbFunction(NbExpression)`

`OutMap = NbFunction [neighbour, neighbour, …](NbExpression)`

where:

`OutMap` is the name of the output map.

NbFunction is a *neighbourhood function* to be used on the values as found by a neighbourhood matrix. Some available functions are: NBMIN, NBMAX, NBAVG, NBSUM, NBPRD, NBSTD, NBMINP, NBMAXP, NBPRDP, and NBCNT. For more information, see Table 9.1.

[*neighbour, neighbour*, …]
if the neighbourhood function is followed by a list of neighbours [*neighbour, neighbour,* …], then the neighbourhood function will only work on the values of these *selected neighbours*.
As in section 9.2.1, *neighbour* is a number between 1 and 9 which indicates the position of a certain pixel in the neighbourhood matrix (refer to Figure 9.3).

(*NbExpression*) is a neighbourhood expression: a neighbourhood expression is any expression which contains the name(s) of the input map(s) and which contains at least one neighbour operator #.

NbExpression may read for instance:

`Map#` a neighbourhood matrix will move over input map `Map`; the values of *all neighbours* found by the matrix will be 'retrieved'. The neighbourhood function will then return the minimum, maximum, average, sum, etc. of these values.

`Map# > 40` a neighbourhood matrix will move over input map `Map`; the values of the neighbours found by the matrix will be 'retrieved' when a neighbour has a value greater than 40. The neighbourhood function will then return the minimum, maximum, average, sum, etc. of these values.

The available neighbourhood functions to perform calculations on multiple neighbours including the central pixel are listed in Table 9.1.

Table 9.1: Overview of neighbourhood functions.

Function	Explanation
NBMIN	Returns the smallest value of the values found by a neighbourhood matrix.
NBMAX	Returns the largest value of the values found by a neighbourhood matrix.
NBAVG	Returns the average value of the values found by a neighbourhood matrix.
NBSUM	Returns the sum of the values found by a neighbourhood matrix.
NBPRD	Returns the predominant value of the values found by a neighbourhood matrix.
NBSTD	Returns the standard deviation of the values found by a neighbourhood matrix.
NBCNT	Counts the number of values as found by a neighbourhood matrix which are not undefined nor false.
NBDIS	Returns the distance between the central pixel and its neighbouring ones.
NBPOS	Returns the position of a neighbour in the neighbourhood expression.
NBMINP	Returns the position of the neighbour with the smallest value.
NBMAXP	Returns the position of the neighbour with the largest value.
NBPRDP	Returns the position of the neighbour with the predominant value.
NBCNDP	Tests for which of the neighbouring pixels a certain condition is true; when none of the neighbours satisfies the specified condition the outcome value is 0.
NBFLT (filter)	Applies a linear filter on the values found by a neighbourhood matrix.

Examples of some simple neighbourhood calculations using a neighbourhood function are:

`OUT = NBMIN(Map1#)` return the minimum value of each 9 values found by a neighbourhood matrix which moves over `Map1`

`OUT = NBAVG(Map1#)` calculate the average of each 9 values found by a neighbourhood matrix which moves over `Map1` (same as using the `AVG3x3` filter in the `Filter` operation).

`OUT = NBCNT(Map1#)` count the number of values which are not undefined of each 9 values found by a neighbourhood matrix which moves over `Map1`.

`OUT = NBAVG[2,4,5,6,8](Map1#)` calculate the average of the values of the neighbours on positions 2, 4, 5, 6, 8 (i.e. the 4-connected pixels) as found by a neighbourhood matrix which moves over `Map1`.

`OUT = NBMINP(Map1#)` return the *position* of the neighbour with the smallest value of each 9 values found by a neighbourhood matrix which moves over `Map1`. The returned position will be a value between 1 and 9.

Neighbourhood function NBFLT which uses a linear filter

NBFLT is always used in combination with *NbExpression*.

`OutMap = `*NbFunction* `(`*NbExpression* `* NBFLT(`*FltName*`))`

where:

`OutMap`	is the name of the output map.
NbFunction	is a *neighbourhood function* (`NBMIN`, `NBMAX`, `NBSUM`, `NBPRD`, `NBMINP`, `NBMAXP`, `NBPRDP`, `NBCNT`, etc.) to be used on the values which result from the neighbourhood expression.
NbExpression	is a neighbourhood expression; a neighbourhood expression is any expression which contains the name(s) of the input map(s) and which contains at least one neighbourhood operator #. In its simplest form, the expression reads: `Map#`.
`NBFLT`	is name of the neighbourhood function which will apply a certain 3x3 linear filter on the values of *NbExpression*.
FltName	is the name of an existing 3x3 linear filter. The existing 3x3 linear filters are: `AVG3x3`, `Edgesenh`, `Laplace`, and `Shadow`. You can also specify a 3x3 filter created by yourself.

An example of a simple neighbourhood calculation using a linear filter is:

`OUT = NBSUM(Map1# * NBFLT(Edgesenh))`

Apply the edge enhancement filter on input map `Map1` and sum the values of each neighbourhood matrix (same as using the edge enhancement filter in the Filter operation).

Calculating a classified slope direction map

In chapter 10 a method is presented to calculate a slope direction map. A simple classified slope direction map (aspect map) can also be made using the neighbourhood function `NBMINP` and a DEM. This function returns the *position* of the neighbour with the lowest value. Because we want all the neighbours to be checked for the lowest value, we use the operator #.

- Type the following formula on the Command line of the Main window:
  `Aspect = NBMINP(Dem#)` ↵

If more pixels have the same minimum value, then the precedence of the pixels is determined by the pixel identifier value (see Figure 9.3): 5, 1, 2, 3, 4, 6, 7, 8, 9. The position of the first pixel with the lowest value in this precedence will be assigned to the pixel of the output map.

The result of the calculation is a raster map with values ranging from 1 to 9 (the position identifiers, see Figure 9.3).

For example, if the neighbour with the lowest value is the lower left (southwest) pixel, the result of the calculation is 7; if the lowest neighbour is the left (west) pixel, the result is 4, etc. The different directions are listed in Table 9.2.

Table 9.2: Relation between the values of the output map and the direction.

Direction	Value
Northwest	1
North	2
Northeast	3
West	4
Flat	5
East	6
Southwest	7
South	8
Southeast	9

- Press Show in the Raster Map Definition dialog box. The map is calculated. The Display Options dialog box is opened. Select Representation Pseudo and click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

Calculating a receiving cell map

Another example of the use of neighbourhood functions is the calculation of a *flow direction matrix*. In a flow direction matrix the direction that a fluid will flow to on a Digital Elevation Model is calculated. This is done by calculating the steepest slope downhill, using the `NBMINP` neighbourhood function in the same way as the direction calculated in the previous exercise. You can however also calculate for each cell from which of its neighbours it receives the fluid, a *receiving cell map*; this map can be calculated using the `10-NBMINP` function.

- Type the following formula on the Command line of the Main window:
  `Flow_direction = 10 - NBMINP(Dem#)` ↵

The result of this calculation is a raster map with values ranging from 1 to 9, indicating from which neighbour the cell receives the liquid. A 1 indicates for example the cell in the Northwest direction. The result can be used in other calculations that predict spreading phenomena (such as the spreading of a pollutant, a landslide, or in runoff modeling). The direction matrix value can be compared with the position identification value, in any 3x3 neighbourhood of which the cell is part. If the pixel value in the direction matrix equals its position number in any 3x3 neighbourhood, then water (or other feature) will flow from this pixel towards the center of this window for which the statement is true.

- Press Show in the Raster Map Definition dialog box. The map is calculated and the Display Options dialog box is opened. Select the Representation Pseudo and click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

Determining flat areas and pits in a DEM

An area is considered to be flat, when in the moving window of 3x3 pixels (9 pixels in total) all pixels have the same value. With the expression `NBCNT(Dem# = Dem)= 9` we can check which pixels have 8 neighbours with the same value as the central pixel. In other words, the statement counts the amount of all neighbours of `Dem` that are equal to the central pixel. If the value is 9, i.e. all pixels *in the matrix* have the same value, the area is considered to be flat. This calculation will result in a `Boolean` map. If the statement is true the result will be 1 else 0.

- Type the following formula on the Command line of the Main window:
  `Flat = NBCNT(Dem# = Dem) = 9` ↵
- Press Show in the Raster Map Definition dialog box. The map is calculated and the Display Options dialog box is opened. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

A so-called pit is a pixel that has a value lower than all of its surrounding pixels. We will check the map `Dem` for the presence of pits, using the following expression: `NBMINP(Dem#)= 5`. In other words, if the central pixel has the lowest value in the matrix, that central pixel will be assigned True (1), or else False (0). This results in a map with domain `Bool`.

- Type the following formula on the Command line of the Main window:
  `Pit = NBMINP(Dem#) = 5` ↵
- Press Show in the Raster Map Definition dialog box. Make sure the map has a `Bool` domain. The map is calculated and the Display Options dialog box is opened. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

This calculation identifies pits in the `Dem` of one pixel. These pits can now be removed by editing the raster map manually or by using some further calculation.

Iterations

Iterations are a special type of calculations. They are a successive repetition of a mathematical operation, using the result of one calculation as input for the next.

The calculation stops when the difference of the output compared to the input is negligible or if the number of iterations as defined before is reached. Iterations can only be used in combination with neighbourhood calculations. Such an application might be for instance the selection of an item or area that fits a certain condition, starting from one pixel (for example the calculation of a flooded area, or the spreading of a pollutant). The iteration functions that can be used are listed in Table 9.3.

Table 9.3: Iteration functions.

MAPITER (*startmap, iterexpr*)	Performs iterations on the start map according to the iteration expression until there are no pixel changes anymore.
MAPITER (*startmap, iterexpr, times*)	Performs a specified number of iterations on the start map according to the iteration expression.
MAPITERPROP (*startmap, iterexpr*)	Performs iterations with propagation on a start map.
MAPITERPROP (*startmap, iterexpr, times*)	Performs a specified number of iterations with propagation on a start map.

Iterations can be performed with or without propagation. When performing iteration with propagation, the newly calculated values for a row of pixels are immediately used as input for the calculation of the next row of pixels. Without propagation, the original values are used as input on all calculations, until the next iteration. After each iteration, ILWIS shows the number of changed pixels. This number is the total number of changes after performing one iteration in all directions (up, down, right and left).

Calculation of the flooded area, given dam site and dam altitude

In this example you are going to calculate the area that will be flooded after the construction of a dam in the mountains, north of Cochabamba, Bolivia.

The first step is determination of the dam site, dam altitude, freeboard and the designed water level *h*. The location of the planned dam is shown in the raster map `Damsite`.

- Display raster map `Damsite`.
- Check the contents of the map.
- After that close the map window.
- Polygonize the raster map `Damsite`. Accept all defaults.

As you can see, the raster map `Damsite` contains the location of the dam, indicated with the word `Damsite`. The rest of the map has undefined values.

Now the Damsite map will be combined with the raster map Dem, and a new map is created in which the Damsite is indicated with the elevation of the top of the dam (3300 meters). The rest of the area shows the elevation of the terrain.

- Type the following formula on the Command line of the Main window:
  Dem_with_damsite = IFUNDEF(Damsite, Dem, 3300) ↵
- Press Show in the Raster Map Definition dialog box. The map is calculated and the Display Options dialog box is opened. Select the Representation Pseudo and click OK.
- You will see that the dam site now has the elevation 3300 meters. Check this by clicking some pixels of the dam. By overlaying the polygon map Damsite it is easier to find the position of the dam.

To determine the area that will be flooded, we need to create a map indicating one pixel in the future reservoir area. Using neighbourhood calculations this pixel acts as the starting point for the calculation. You will use the pixel editor to create the new raster map.

- From the File menu in the map window, select Create, Raster Map. The Create Raster Map dialog box is opened.
- Type for raster Map Name: Start.
- Select as GeoReference: Cochabamba.
- Select Domain: Value with a Value Range of 0 to 1, and a Precision of 1. Click OK.
- The Pixel editor is opened; on the Status bar a message appears: Please zoom in to edit. Zoom in on the dam site until you see the individual pixels.
- Select a pixel just *upstream* of the dam site, press the right mouse button and select Edit from the context-sensitive menu. Type the value 1, and press Enter.
- Press the Exit Editor button. Now you have a raster map Start with 1 pixel that has the value 1. All other pixels are undefined.
- Close the map window.

Now you can perform the actual calculation of the flooded area. Iteration with propagation is used until there are no changes anymore in any of the pixel values. You can perform iterations by including the iteration statement in a Map Calculation formula. You can also use the dialog box of the Iteration operation. Since the top of the dam has an elevation of 3300 meter and the freeboard of the dam is 20 meter, the *actual* water level upstream the dam has an elevation of 3280 meter.

- Double-click Iteration in the Operation-list. The Map Iteration dialog box is opened.
- Select the Start Map: `Start`.
- In the Expression box type:
  `IFF(Dem_with_damsite > 3280,Start,NBMAX(Start#))`

This means: if the altitude in the new Digital Elevation Model is more than 3280 meter, then return the pixel values of raster map `Start` (which are undefined). Otherwise, assign the maximum value of the neighbouring pixels found in raster map `Start` (which is a value of 1). In the first iteration there is only one pixel that has value 1 (the starting pixel). In every iteration, the neighbouring pixels that satisfy the conditions (altitude < 3280 meter) will get the same value as that starting pixel. This will continue until the next neighbouring pixels have an altitude of more than 3280 m. In all areas lower than 3280 meter *other* than upstream the dam site, there is no start pixel defined. Therefore the neighbours will always be undefined and therefore the areas will also stay undefined.

- Accept the default value for the Stop Criterium: `Until No changes`. This means that the iteration continues until no changes occur anymore.
- Make sure that the option Propagation is selected, so that the new pixel value is immediately used in the calculation of the next line.
- Type for the Output Raster Map: `Flooded`.
- Select the Domain `Value`, the Value Range `0` to `1` and the Precision of `1`. Click the Show button.

The calculation may take a minute. The program will calculate in a downward, upward, left and right direction. When no changes occur after a full iteration, the final map is generated, and the Display Options dialog box is displayed.

- Accept the defaults and click OK.
- Use the pixel information window and add the maps `Dem_with_damsite`, `Start` and `Flooded`. Check the results.
- Close the pixel information window and the map window.

The output map `Flooded` can now be used to calculate the volume of the water in the reservoir.

- Cross the raster map `Flooded` with the raster map `Dem_with_damsite`. Name the Output Table `Volume` and click Show.
- In the table, calculate the water level with the formula:
  `Difference = 3280 – Dem_with_damsite`
- In the Column Properties dialog box accept the defaults and click OK.
- Calculate the volume of the area with the formula:
  `Volume = Difference * Area`
- In the Column Properties dialog box accept the defaults and click OK.

What is the total volume of the flooded area?
Is this dam well situated? Why?

The concept of map iterations is very powerful. It allows you to create your own models for different types of applications. For more examples of neighbourhood calculations, consult the ILWIS Applications Guide and the topic How to calculate (advanced) in the ILWIS Help.

Summary: Neighbourhood functions

- Neighbourhood calculations are a special type of spatial analysis in ILWIS. They are calculations on pixels in which the outcome depends on the neighbouring pixels.

- Neighbourhood calculations may be performed on user-selected pixels as well as on whole maps.

- Just as in filtering procedures, neighbourhood calculations make use of a matrix. This imaginative window of 3x3 cells is moved over the raster map. Each cell of the output map is calculated according to the specified neighbourhood expression.

- Neighbourhood calculations can make use of neighbourhood functions.

- Iterations are a special type of calculation. They are a successive repetition of a mathematical operation, using the result of one calculation as input for the next.

- In iterations, the calculation stops when the difference of the output compared to the input is negligible, or if the number of iterations as defined before is reached.

- Iterations are always used in combination with neighbourhood calculations. Such an application might be for instance the selection of an item or area that fits a certain condition, starting from one pixel (for example the calculation of a flooded area, or the spreading of a pollutant).

9.3 Distance calculation

The Distance operation calculates the distance (in meters) from user specified source pixels to all other pixels in a raster map. The result of this operation is a raster map, where every pixel is assigned a value representing its distance to the nearest source pixel. The user-defined source pixels from which the distance is calculated may represent any kind of features, such as point features (for example rainfall stations), linear features (for example roads) or area features (for example a city block). The distance is calculated with a distance filter. The principle is shown in Figure 9.4.

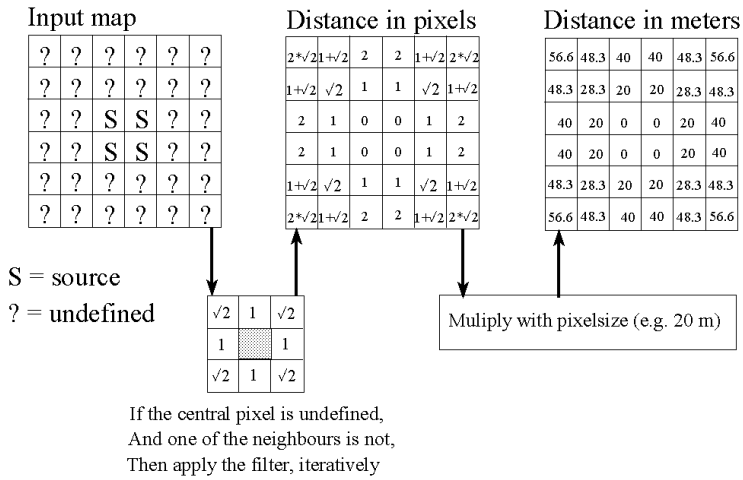


Figure 9.4: The principle of distance calculation.

A *source map* for distance calculation can be any raster map with domain type class, ID or value. This map should contain pixels with class names, IDs or values and pixels with undefined values. All pixels with a class name, ID, or value in a source map, are considered as source pixels and distance values will be calculated for all pixels that are undefined. The distance filter will first calculate the distance in number of pixels. The distance from a source pixel to its horizontal or vertical neighbours is 1, and the distance from a source pixel to its diagonal neighbours is the square root of 2. The filter will move first from the first pixel on the top line to the last pixel of the last line. Then, it will do the same operation in the reverse direction. Once the distance, in number of pixels, to the nearest source pixel is known, the final distance in meters is obtained by multiplying this with the pixel size.

If not all parts in the source map are equally accessible, a so-called *weight map* can be used. A weight map contains pixels with values (weight factors) referring to the degree of accessibility and inaccessibility. In a weight map, accessible pixels are assigned positive values. A higher positive weight value for accessible pixels means that the area covered by these pixels (swamps, forest, bad roads, etc.) are more difficult to cross than other areas. Inaccessibility can be introduced in the weight map by negative weight factors. Pixels with a negative value (river, dens forest or

impenetrable areas), represent those areas in the map which can not be passed. No distance value will be calculated for these pixels; the output value will be undefined.

By default, the output map (distance map) obtains the system `Distance` domain. The value range and step size of the output map can be adjusted each time you perform the Distance operation. You can also create or use a value domain of your own.

If line barriers, like a river, are used with a width of only one pixel (rasterized segments), these barriers should be broadened before performing the distance calculation, e.g. manually in the pixel editor or automatically with a `Dilate` filter or the `Conn8to4` filter (see the Filtering exercises in section 9.1). This because the distance program can pass diagonal barriers of one pixel wide, as it uses 8-connected distances (meaning each pixel has access to its eight neighbours: horizontally, vertically, and diagonally).

In the following exercise three examples of distance calculations will be shown: one from line features (drainage), one from area features (city blocks, taking into account inaccessible areas) and one from point features (rainfall stations, using Thiessen polygons).

Simple distance calculation

The map `Drainage` with a class domain is used in order to calculate the distance from streams and lakes to any pixel in the map, under the assumption that all areas are equally accessible. The raster map `Drainage` contains pixels, representing the drainage (source pixels) and undefined pixel, representing the rest of the area in the map.

- Double-click the operation Distance Calculation in the Operation-list. The Distance Calculation dialog box is opened.
- Select raster map `Drainage` in the list box Source Map.
- Type `Drain_distance` in the text box Output Map.
- Accept the other defaults and click the Show button.

The program will calculate first in a forward direction (starting from first pixel in the first line to the last pixel in the last line), after which it calculates in the backward direction. After a while the Display Options – Raster Map dialog box is displayed.

- In the Display Options - Raster Map, select Representation `Clrstp12`, change the Stretch range into `0` to `2000` and click OK.

The distance map is displayed on the screen. This map is a value map, in which the pixel values refer to the distance of this pixel towards the nearest source pixels. You

can also display the segment map Drainage in the same map window.

- Drag the segment map Drainage to the map window.
- Click OK in the Display Options dialog box.
- Enlarge the map and check the distance values by clicking on several pixels.
- Close the map window.

Calculating distance with weights: Travel time map

A travel time map is a distance map in which the calculated distance values in meters are converted to time units (seconds, minutes, or hours, etc.). For this process we need two raster maps as input: a source map and a weight map. The source map indicates from where the distance should be calculated, and the weight map indicates which areas are more difficult to cross.

The following examples show how to convert distance values to time units, assuming a maximum speed of 20 km/h:
- When the output distance values are divided by 20,000 (m/h), the time in hours to reach the nearest source pixel is obtained.
- When the output distance values are multiplied by 60 and divided by 20,000, the time in minutes to reach the nearest source pixel is obtained.
- When the output distance values are multiplied by 60*60 and divided by 20,000, the time in seconds to reach the nearest source pixel is obtained.

In this exercise, the city block map of Cochabamba (Cityblock) is used to calculate how much time it takes from any part in the city to reach the central plaza Plaza 14 de Septiembre. For reasons of simplicity we assume that you can travel equally fast in all the streets of the city. The city blocks themselves cannot be crossed, with the exception of those with a recreational use (parks, etc.), which can be crossed with a maximum velocity of 5 km/h.

First step: Creation of the source map

One of the input maps required for this operation, is a source map in which only the central plaza has a value and the rest of the map has undefined values. The central plaza has the ID 018 (you can check that if you like).

- Type the following formula at the Command line in the ILWIS Main window:
  `City_source = IFF(Cityblock = "018", Cityblock, "?")` ↵
- Click Show in the Raster Map Definition dialog box.
- Check the results in map City_source and close the map afterwards.

Second step: Creation of the weight map

The second input map required for this operation, is a weight map where all the pixels, except the source pixels, are assigned *weight factors* to simulate the difficulty of crossing pixels, which form barriers such as building blocks, the river, the lake, mountain ridges, etc. In this exercise the roads are assigned a value 1, the recreational areas a value 4 and the rest of the city blocks will receive the value -1 (negative weights make the pixels inaccessible). Note that the weight factors are inversely proportional to the speed. In this exercise the maximum speed (20 km/h) is set to weight factor 1 (along the streets) and to obtain the minimum speed of 5 km/h in the recreational areas we use a weight factor 4.

- Type the following formula at the Command line of the ILWIS Main window:
  `City_weight = IFUNDEF(Cityblock, 1, IFF (Cityblock.Landuse = "Recreational", 4, -1))` ↵
- Click Show in the Raster Map Definition dialog box.
- Check the results in map `City_weight` and close the map afterwards.

In this formula, first the roads (which have undefined values) are assigned a value `1`. Then, if the land use type (in the column `Landuse` of the table `Cityblock`) is "`Recreational`", the resulting value is `4`. All other pixels (the other city blocks) will get a value `-1`. Note that these type of calculations only work when the attribute table of a map is connected to that map, i.e. that in the property form of the map the option `Attribute Table` is switched on and the correct table is selected in the list box.

Third step: Calculating distances

- Expand the Raster Operations item in the Operation-tree and click the Distance Calculation operation.
- In the Distance Calculation dialog box select raster map `City_source` in the list box Source Map.
- Click the check box Weight Map and select raster map `City_weight`.
- Type `City_distance` in the text box Output Raster Map.
- Accept the other defaults and click the Show button.

The distance calculation takes quite some time. The program calculates forward (from left to right and from the first line to the last line) and backward (the reverse). The tranquilizer shows the number of changes that are made, as well as the line that is being calculated.

The calculation is done iterative until there are no more changes. When the calculation is finished, the Display Options - Raster Map dialog box is displayed.

- Select Representation Pseudo and click OK. The weighted distance map is displayed on the screen.
- Find out the distance from the airport to the city center.

Fourth step: Converting distances to travel time

Now that the weighted distances from each part in the map towards the central plaza are known, we only need to convert the distance values (in meters) to time units (minutes). The maximum speed in the city is assumed to be 20 km/h in the streets and 5 km/h in the parks. To create the travel time map:

- Type the following formula on the Command line:
  `City_travel_time = (City_distance * 60 / 20000)` ↵
- Click Show in the Raster Map Definition dialog box. Select Representation Pseudo and click OK.
- Check how much time it takes to travel from the airport to the center.
- Close the map window when you are finished.

Calculating distances: Thiessen map

In the third and last example of distance calculation, you will have a look at a *Thiessen map*. In a Thiessen map each pixel is assigned the class name, identifier, or value of the nearest point. For example, schools, hospitals, water wells, etc. can be represented by points. 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 horizontal distance between input points and output pixels.

In this exercise a Thiessen map will be calculated from point map `Rainfall`, containing - fictitious- rainfall stations in the Cochabamba area. Before we can calculate it, we first need to rasterize the point map `Rainfall`.

- Click with the right mouse button on point map `Rainfall` and select Rasterize and the Point to Raster command from the context-sensitive menu. The Rasterize Point Map dialog box is opened.
- Select GeoReference `Cochabamba`, name the Output Raster Map `Rainfall` and click Show.
- Click OK in the Display Options - Raster Map dialog box. The map is now displayed. Since each point is represented by one single pixel the map appears to be empty. Only if you zoom in deeply on a point location, you can see the rasterized points.
- Close the map.

➥

- Double-click the Distance Calculation operation in the Operation-list. The Distance Calculation dialog box is opened.
- Select raster map Rainfall in the list box Source Map.
- Type Rain_distance in the text box Output Raster Map.
- Select the check box Thiessen Map and enter the name: Rain_area.
- Click the Show button.
- After the calculations, the Display Options – Raster Map dialog box is opened.
- Click OK to display the output map Rain_distance.
- Also display the Thiessen map Rain_area, and position it next to the map Rain_distance.
- Click the various units to find out their meaning.

The Thiessen map shows the areas, which are nearest to each one of the rainfall stations. We can also display the attribute data (rainfall values) instead of the rainfall stations. We will first calculate the total rainfall per year for each of the stations.

- Open table Rainfall.
- Type the following formula on the Command line of the table window:
  Rain_total = January + February + March + April + May + June + July + August + September + October + November + December ↵
- The Column Properties dialog box is opened. Click OK.
- Close the table window and activate the map Rain_area.
- Re-open the Display Options – Raster Map dialog box of the map.
- Select the check box Attribute and select the column Rain_total.
- Select Representation Pseudo and click OK. Now the total rainfall per year in shown for each of the Thiessen areas.
- Close the maps Rain_distance and Rain_area.

The Thiessen map can be used to calculate the total rainfall per year for each catchment of the map Catchment.

- Calculate the annual rainfall per catchment. First use the Attribute Map of Raster Map operation to generate a map Rain_total from the Thiessen map Rain_area, and use column Rain_total from the attribute table Rainfall.
- Use the Cross operation to overlay the maps Catchment and Rain_total.

- Calculate the total rainfall per catchment in the cross table with the Aggregation Function Sum in the table window.
- When you have finished, close all the map and table windows that are open.

A Thiessen map can also be created, using the Nearest Point interpolation method (see chapter 11).

Summary: Distance calculation

- The Distance operation calculates the distance (in meters) from user specified source pixels to all other pixels in a raster map.

- All pixels with a class name, ID, or value in a *source map*, are considered as source pixels and distance values will be calculated for all pixels that are undefined.

- A weight map contains pixels with values referring to the degree of accessibility. Negative weights indicate inaccessibility, and positive values, higher than 1, indicate areas that can be crossed with more difficulty.

- The units in a distance map are meters. They can be converted with Map Calculation formulas into time.

- A by-product of distance calculation is a Thiessen map, in which the 'service area' of source pixels, based on the shortest distance is indicated. Thiessen maps can also be made from point maps using the Nearest Point interpolation.

9.4 Area numbering

The Area Numbering operation assigns *unique identifiers* to class, value or ID pixels in a raster map; the identifier `Area` followed by a unique number starting at `1`.

Area numbering may be a useful tool in the following situations:
- If you want to perform a Cross operation with another map and you want to distinguish each connected area separately, instead of using mapping units.
- If you want to add special attribute data to each connected area, instead of the mapping units.
- To generate maps that can be used as input for cell based models.

The operation can be used on connected (4- and 8-connected) and not connected pixels; undefined pixels are ignored. Pixels from which all neighbours are undefined are considered not to be connected. Connected pixels that have the same class are assigned the same identifier.

Not connected pixels, connected pixels with a different class than its neighbour and ID pixels are assigned a unique identifier.

The operation is based on a moving 3x3 matrix that assigns an ID to the central pixel. The window starts at the first row and moves from left to right. The first (connected) pixel(s) encountered will get the identifier `Area 1`, the second `Area 2` etc. Furthermore, an attribute table is created, which contains the new area IDs and the original class names, IDs or values.

It is not recommended to use a value map as input for this operation; this can result in an output map where every pixel has its own identifier. The procedure to follow with value maps is to first classify the value map to create the classes that you want to number and then use the output class map as input for the area numbering operation.

Figure 9.5 shows a simple example of the area numbering operation applied to a bit map.

In this exercise, the Area Numbering operation is performed on map `Geomorphology`, which has a class domain.

- Double-click the Area Numbering operation in the Operation-list. The Area Numbering dialog box is opened.
- Select class raster map `Geomorphology` in the list box Raster Map.
- Type `Geomorphology_numbered` in the text box Output Raster Map.
- Accept the defaults (note that you use the 8-connected option) and click the Show button. The Display Options - Raster Map dialog box is opened.

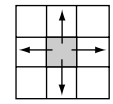
- Select the option Multiple Colors 31 and click OK. The map is displayed.
- Check the meaning of the mapping units. Double-click a unit. The Edit Attribute dialog box is opened, in which you can also read the geomorphologic class name.
- Close the map window when you finished the exercise.

Input Map:

A	?	?	?	?	A
?	A	A	?	A	A
A	?	A	?	A	A
?	?	?	?	?	?
A	A	A	?	A	A
A	A	?	A	A	A

Result: 4-connected

1	?	?	?	?	2
?	3	3	?	2	2
4	?	3	?	2	2
?	?	?	?	?	?
5	5	5	?	6	6
5	5	?	6	6	6



Result: 8-connected

1	?	?	?	?	2
?	1	1	?	2	2
1	?	1	?	2	2
?	?	?	?	?	?
3	3	3	?	3	3
3	3	?	3	3	3

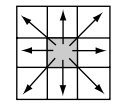


Figure 9.5: Simple example of Area numbering. The pixels that are connected are assigned the same code. Different results are obtained when only the horizontal and vertical neighbours are considered (4-connected) or whether all neighbours are considered (8-connected).

Summary: Area numbering

- The objective of area numbering, is to assign *unique identifiers* to those areas in a raster map that are having the same value, and that are connected.

- Area Numbering is similar to the Unique ID operation that can be used for point, segment and polygon maps.

9.5 Connectivity calculations

Connectivity calculations look at spatial units that are connected (using a set of pre-defined rules). These spatial units may either occur in raster or vector maps. This group of calculations can be subdivided into:
- *Contiguity functions*: Connected areas that share (a) common characteristic(s);
- *Proximity functions*: Connected areas having the same distance (in time, distance, costs, etc.) to a given point, line or area;
- *Network functions*: Areas (usually lines) that form a set of interconnected features through which resources from one location to another are moved;
- *Spread functions*: Connected areas that result for spreading, dilution or accumulation of phenomena from a given point, line or area, and;
- *Seek functions*: Connected areas (usually lines) that form an optimum pathway according to (a) specified decision rule(s).

Contiguity calculations

Contiguity calculations examine the connected areas that share (a) common characteristic(s). In order to know which areas are connected, topological information is required. In general, these calculations are applied to identify (retrieve) areas with a specific size and with specific characteristics.

In this exercise we will show you how you can use the information of neighbouring polygons to solve the following problem:
Find out a forested area which is not bordering agricultural land or the city, and which has a closed area of at least 200 hectares.

In order to solve this problem we need to know the following:
- Which one of the forested areas has no borders with agricultural or urban areas?
- Which one of the forested areas is at least 200 hectares large?

In order to solve these two sub problems we need to work with the polygon map `Landuse`. The `Landuse` map has a class domain. This means all the polygons with the same land use have the same class. However, we cannot work on mapping units. We have to know the information for each individual polygon. In order to do so we apply the operation Unique ID. This operation will transform a class map into a unique identifier map, in which each polygon receives a single code.

- Double-click the Unique ID operation in the Operation-list. The Unique ID dialog box is opened.
- Select polygon map `Landuse` as the Input Map.
- Type for Domain Prefix `LandID`.
- Type `LandID` as the Output Map.
- Type the following Description: `Unique Landuse polygons`.
- Click Show. The map is being calculated.

- Click OK in the Display Options - Polygon Map dialog box. The map is now displayed.
- Find out the codes of the map by clicking on the units.
- Double-click on a unit. Now the land use type for each polygon is displayed in the Attributes window.

Apart from the map `LandID`, a table with the same name is also generated in which the land use type for each polygon is stored. The next thing we need to know is which are the neighbouring polygons of each polygon. For this you can use the operation Neighbour Polygons. This operation will calculate for each polygon in the map, what its neighbours are, and what the length of their boundary lines is.

- Close the map `LandID`.
- Double-click the Neighbour Polygon operation in the Operation-list. The Neighbouring Polygons dialog box is opened.
- Select the polygon map `LandID` as Input Map.
- Type `LandIDnb` as Output Table.
- Type the following Description: `Neighbours of the unique land use polygons`.
- Click Show. The table is opened after it has been calculated.

The result of this operation is a table (`LandIDnb`) containing three columns:
- `PolName1`: The polygon for which the neighbours are calculated.
- `PolName2`: The polygon neighbouring the polygon shown in `PolName1`.
- `Length`: The length of the boundary line between the two polygons.

The combination between the polygons in `PolName1` and those in `PolName2`, represent the polygons located at the margins of the map. The undefined values (?) refer to the boundary of a polygon located at the margin of the study area.

We are interested in the polygons with the land use type `Forest` in the column `PolName1`. As you can see, we do not have information yet on the land use types (only on the polygon ID's). This information exists, however, in the table `LandID`, in which for each ID the land use is given. We will have to join the two tables in order to read the information of the land use in table `LandIDnb`.

Joining of tables was already discussed extensively in chapter 5.

- In the table open the Columns menu and select the Join command. The Join Wizard appears.

Since the column `PolName1` has the same domain as the table `LandID`, the joining is fairly simple.

- Select the Table : LandID.
- Select the Column: Landuse and click the Next button.
- The Key-Column option is selected by default. Click the Next button.
- Select as Key 1: PolName1 and click the Next button.
- Type the Output Column: Landuse1 and click the Finish button.
- Click OK in the Column Properties dialog box.

Now we know for each of the polygons ID's in column PolName1 what the land use is. We also need to know the land use of their neighbouring polygons, listed in column PolName2.

- Do the same Join operation, but now select the Key-Column: PolName2 and type the Output Column: Landuse2.

Now that we know the land use type of each polygon and its neighbours, we can start to evaluate which one of the forest polygons has no borders with agricultural or urban polygons. We can do this with a calculation formula. There are four types of land use which we want to exclude: Agriculture, Agriculture (irrigated), Urban center and Urban periphery. To simplify the formula, we already know that none of the forested area is bordering the urban area. The formula would become very large if we would have to type the full names. Fortunately we can use a shortcut. If we use the function Left(column, length) in the following way:
- Left(Landuse2, 2) = "Ag", this will incorporate both the land use types Agriculture and Agriculture (irrigated).

- Locate the mouse pointer on the Command line of the table window and type (case sensitive!) the following formula:
  Good = (Landuse1 = "Forest") and (left(Landuse2, 2) <> "Ag") ↵
- Click OK in the Column Properties dialog box.

The column Good has a Bool domain. You know now all the polygons with the land use type "Forest", that are not bordered by agricultural or urban polygons. These are classified as "True" in the Good column.

- Open the Columns menu and select Sort. The Sort dialog box appears.
- In the Sort dialog box select the Column Landuse1 accept the other defaults and click OK.
- Scroll down until you see the records with the land use type "Forest" in the column Landuse1.
- Check the result of the formula.

As you can see there are 3 forest polygons which are bordered by agriculture. Another 13 are bordered by either grassland or shrubs, and one forest polygon is located at the margin of the area. However, these records may refer to the same forest polygons, which may be bordered by agriculture on one side and grassland on the other. Therefore, we should take the minimum value of the column Good in the next step.

Now we know which of the forest polygons is not bordered by agricultural or urban land.

- Close the table LandIDnb and open the table LandID.
- Open the Columns menu and select the Join command. The Join Wizard is opened.
- Select the Table LandIDnb and select the Column Good. Click the Next button.
- Accept the default option Use Domain of current table… and click the Next button.
- Select Polname1 as Key 2 and click the Next button.
- Select the Function Minimum and click the Next button.
- Type the name of the Output Column Good and click Finish.
- Click OK in the Column Properties dialog box.

Now the table LandID contains the column Good, that indicates with "True" those polygons that have forest, and have a neighbour polygon which is not agriculture nor urban.

The calculation for the first requirement is finished. Now we still need to know which of the forest polygons has an area larger than 200 hectare. We can obtain this information from the polygon histogram.

- Click with the right mouse button on the polygon map LandID and select Statistics, Histogram from the context-sensitive menu. The Calculate Histogram dialog box is opened.
- Click OK. The Polygon Histogram is shown as a table.

Now that the area of the polygons is known, we can make the final evaluation.

- Close the Polygon Histogram LandID.
- Open or maximize the table LandID.
- Type the following formula on the Command line of the table window:
  `Finalgood = Good and (LandID.hsa.area > 2000000)` ↵
- Click OK in the Column Properties dialog box.

The calculation uses the column Area from the polygon histogram table (extension .hsa). Those polygons that contain the word "True" in the column Finalgood, fulfill the requirements stated in the beginning. They are forest polygons, with an area of at least 200 hectares, and not bordered by agriculture or city polygons.

The final task is to display these polygons.

- Close the table LandID.
- Double-click the polygon map LandID. The Display Options dialog box is opened.
- Select the check box Attribute and select the Column: Finalgood.
- Click OK. The polygon map is displayed and you can see the suitable polygons displayed in green.
- Check the result by clicking on these polygons and close the map window afterwards.

Summary: Connectivity calculations

- Connectivity calculations look at spatial units that are connected (using a set of predefined rules).

- The operation Unique ID is used to transform a class map into a unique identifier map, in which each polygon receives a single code.

- The operation Neighbour Polygons calculates for each polygon in a map, what its neighbours are, and what the length of their boundary lines is.

- The resulting table can be used for connectivity calculations.