Geography 4203 / 5203

# GIS & Spatial Modeling

**Class 4**: Raster Analysis I

# Some Updates

- **Readings discussions**

  Reminder: Summaries!!! Submitted BEFORE discussion starts…

- **Labs next week:**

  Start (delayed) of lab season as scheduled (M / W 9-11am)

# Last Lecture

- We talked about **raster data** as one form of **tesselation**, their properties and important things you will find when working with them
- These facts are important to understand **storage** limitations, **data types**, **bit depths** and **resolution-related** problems
- In this context the **assignment** of pixel values in classifications or vector-raster conversions can become complex and is crucial

# Last Class Meeting

- Right, and remember the last session which was a readings discussion about the **field-object** debate
- Take some impressions with you from this nice session - we did not come to an end (nobody ever did)
- Keep in mind the different **conceptual model** approaches and their counterparts in different disciplines
- Try to remember what the advantages of field representations are in **modeling**, **error analysis** and **mapping**

# Today‘s Outline

- We will talk about some important basics of **raster analysis** and **Map Algebra**
- You will hear about the **principles** of Map Algebra as the foundation of **GIS modeling framework**
- We will talk about the single **building blocks** of this **modeling language**: operators, functions, parameters and objects
- This will give you an impression of the **concepts** behind it

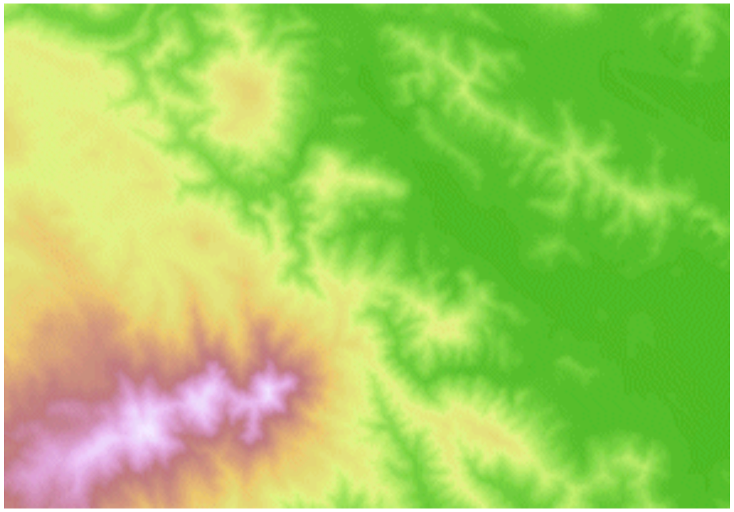
# Learning Objectives

- You will learn what **Map Algebra** stands for and what the single components are
- You will have insights into the **functioning** of Map Algebra
- You will understand what **functions** you are going to use, what an **operator** is and how to combine all elements for **iterative** models under **control**

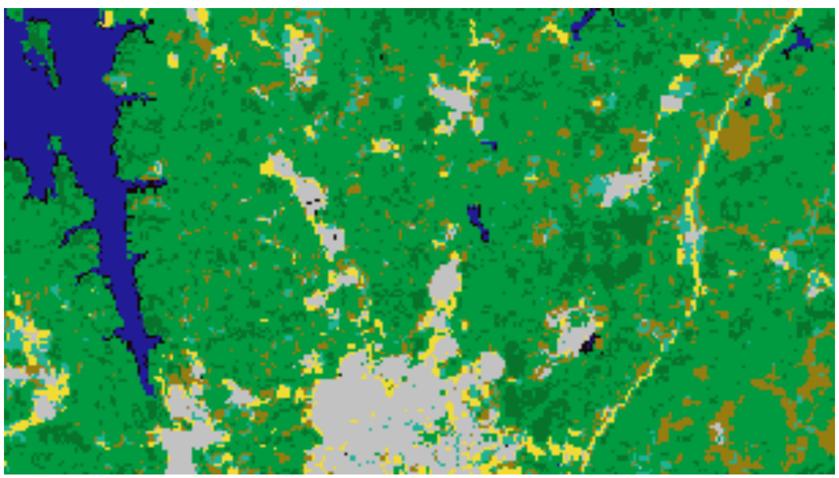
# Some Repitition

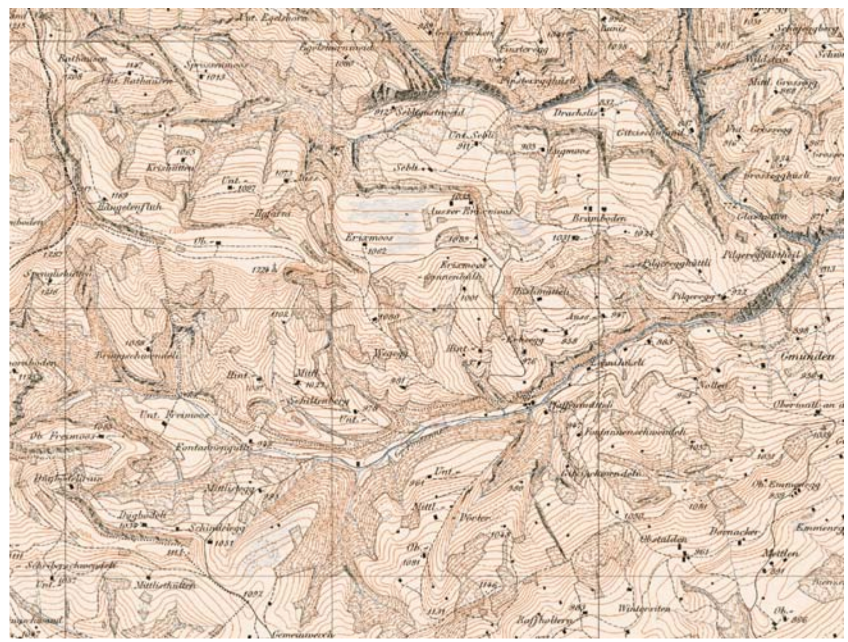

| binary | decimal |
|---|---|
| 00000001 | 1 |
| 00000010 | 2 |
| 00000011 | 3 |
| 00000100 | 4 |
| 00000101 | 5 |
| 00000110 | 6 |
| 00000111 | 7 |
| 00001000 | |
| 00001001 | |
| 00001010 | |
| 00001011 | |
| .... | |

| Bit depth | Range of values that each cell can contain |
|---|---|
| 1 bit | 0 to 1 |
| 2 bit | 0 to 3 |
| 4 bit | 0 to 15 |
| Unsigned 8 bit | 0 to 255 |
| Signed 8 bit | -128 to 127 |
| Unsigned 16 bit | 0 to 65535 |
| Signed 16 bit | -32768 to 32767 |
| Unsigned 32 bit | 0 to 4294967295 |
| Signed 32 bit | -2147483648 to 2147483647 |
| Floating-point 32 bit | -3.402823466e+38 to 3.402823466e+38 |




Flaz
Flaz
Pontresina
Ova da Bernina





# World Files for Georeferencing Information

- Some image formats store GI in a header of the image file (grids, img, GeoTIFF)
- Others use world (ASCII) files (.tfw)
- Origin of an image is ul (row values increase downward), of a coord system ll

```
20.17541308822119 - A
0.00000000000000 - D
0.00000000000000 - B
-20.17541308822119 - E
424178.11472601280548 - C
4313415.90726399607956 - F

A = x-scale; dimension of a pixel in map units in x direction
B, D = rotation terms
C, F = translation terms; x,y map coordinates of the center of the upper left pixel
E = negative of y-scale; dimension of a pixel in map units in y direction
```

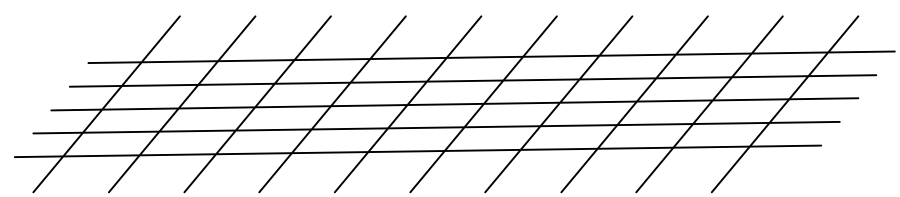
# Critical Points in Working with Raster Data

- Noise, false colors, mixed colors
- Object separation / identification
- Neighborhoods for Morphology operators
- Assignment and coding
- Edges, contours and transitions between objects and background (blurring)

# Raster and Raster Analysis

- **Two-dimensional arrays** organized in columns and rows as basis for efficient computation (translation into code)
- Making use of the **simple** and **flexible** data structure
- Fields, objects, regions, connected components, networks
- **Developing** and **extending** functionalities and operators

```
MyRaster[row][col]
```



# Understanding Raster Analysis

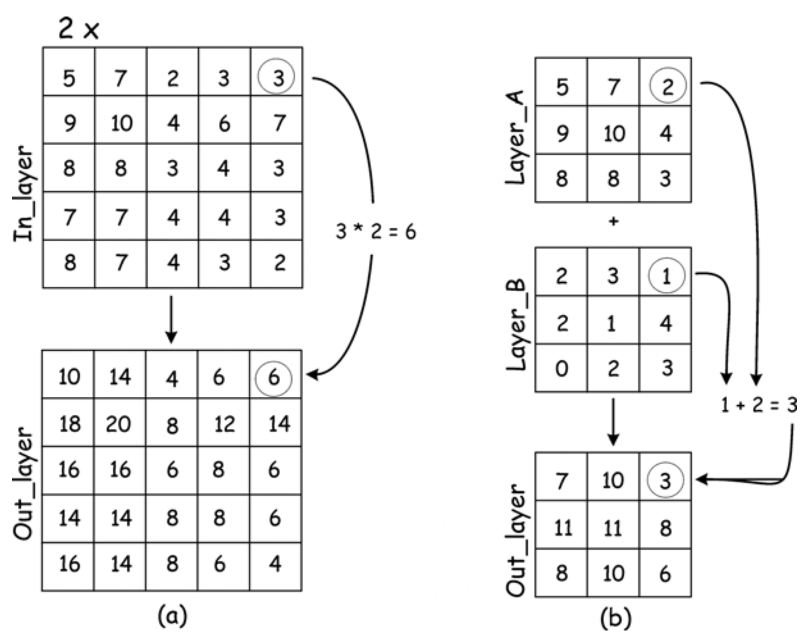
- To apply **analysis tools** is easy and straightforward
- Behind existing tools are complex **algorithms** implying mathematics, geometry and matrix operations between datasets and within regions of the same layer
- Thus a basic **understanding** of the concepts behind these tools is fundamental

# Map Algebra

- Introduced by Dana **Tomlin** and Joseph **Berry**
- **Cell-by-cell** combination of raster data layers (addition, subtraction, multipl.,…)
- Simple **operations** on numbers stored as values at **raster** cell locations
- Output grids with results at the cell locations **corresponding to input cells**

# Map Algebra Operations

- **Unary** Operations
- **Binary** Operations
- **Higher-order** Operations



# Map Algebra and Matrix Algebra

- **One-to-one** locational correspondence throughout all functions applied
- *, /, **, root are defined by the same rules of maintaining the one-to-one translation
- **Matrix Algebra** would apply rules for mathematical matrices (do you remember them?)
- If A is a 2x3 and B is a 3x2 matrix:

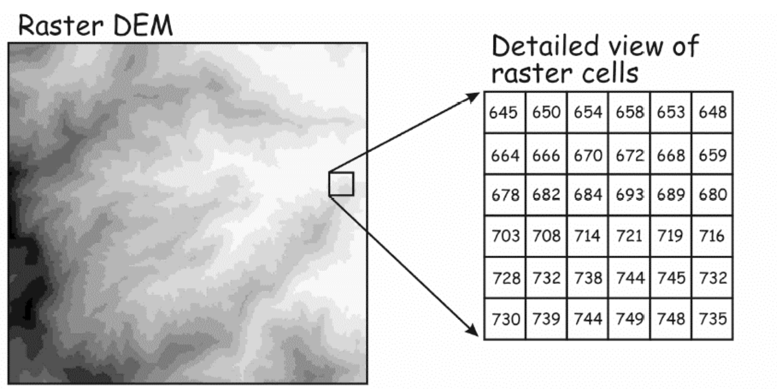
$$\mathbf{AB} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix}$$

$$\mathbf{BA} = \begin{bmatrix} b_{11}a_{11} + b_{12}a_{21} & b_{11}a_{12} + b_{12}a_{22} & b_{11}a_{13} + b_{12}a_{23} \\ b_{21}a_{11} + b_{22}a_{21} & b_{21}a_{12} + b_{22}a_{22} & b_{21}a_{13} + b_{22}a_{23} \\ b_{31}a_{11} + b_{32}a_{21} & b_{31}a_{12} + b_{32}a_{22} & b_{31}a_{13} + b_{32}a_{23} \end{bmatrix}$$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{p1} + b_{p1} & a_{p2} + b_{p2} & \cdots & a_{pn} + b_{pn} \end{bmatrix}$$

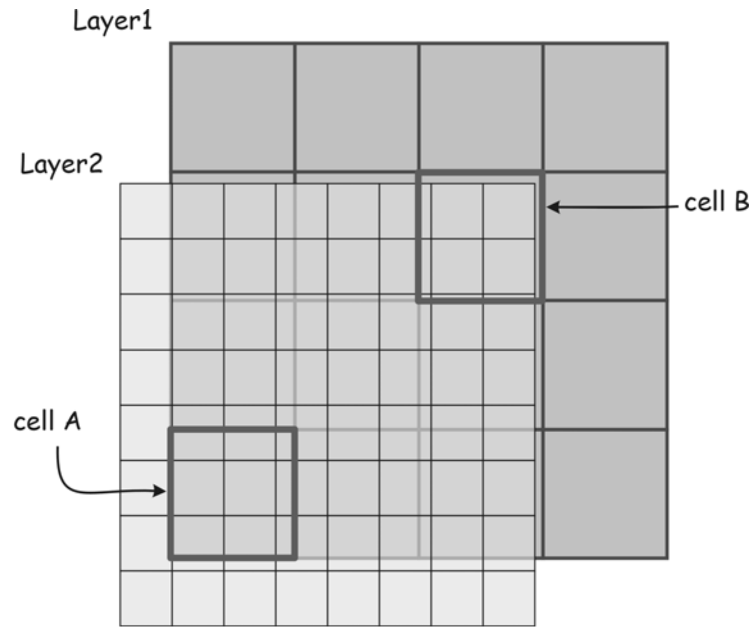
# Why we are not doing Matrix Algebra

- So that is the special sense of Map Algebra:
- **Position** of individual grid cells corresponds to their position in **geographic space** (not in math matrices)
- Cell values are changed but **not** their position…
- Intuitively easy to get but it is important to know these differences… and **why** they exist



# Raster-Related Problems

- Different **extents** and ***NoData*** values (the common or “shared” area?)
- Different raster **cell sizes** (how to define comb.)
- **Misalignment** of raster cells




- **Resampling** & **Transformations**

# Map Algebra as Modeling Language

- Something like a modified version of Matrix Algebra, yes!
- But it can be seen as a **complete modeling language** (taken as standard for industry)
- Allows for program **control**, **development** and **iteration** + mathematical manipulation and logical **operators** of comparison
- …and thus for the **whole complexity** of modeling
- Fuzzy logic as one example

# Rules to be Followed

- E.g. in ESRI's Spatial Analyst
- **Building blocks** for Map Algebra language are:

  **Objects** - datasets, layers, values (as inputs or storage location)

  **Actions** - performed on objects; *operators* and *functions* (loc,foc,zon,glob) + *application functions*

  **Qualifiers on the actions** - parameters determining the conduction of a function

# Actions I: Basic Operators

- Computations **within** & **between** objects
- Remember **NoData** values…
- **Arithmetic** (+,-,*,/,**,root,mod)
- **Relational** / Compare / Conditional (<,>,==,>=,…)
- **Boolean** / Logical (&&, ||, !, XOR)
- **Combinatorial** (and, or; unique identifiers not Boolean)
- **Assignment** (=)
- **Accumulative** (+=, -=,…)
- **Logical** (IN, DIFF, OVER)

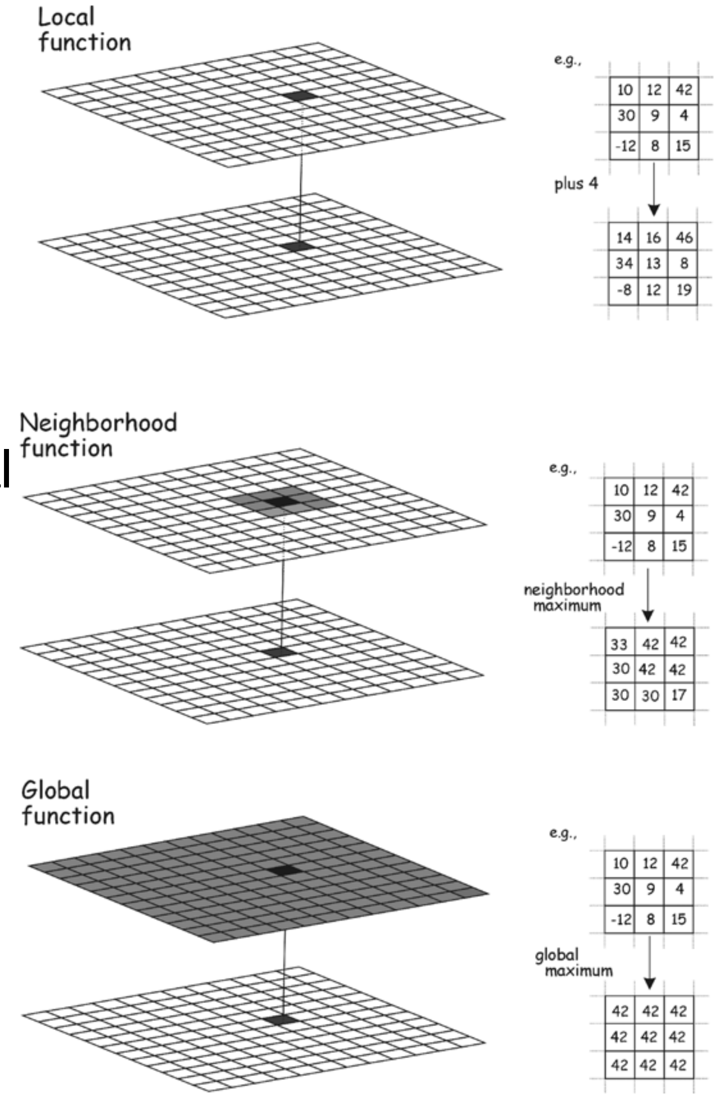
# Actions II: Functions

- Spatial **modeling** tools for cell-based data
- **Higher-order** GIS operations
- Important building blocks for model **implementation** based on Map Algebra
- **Parameter-dependent**

# Function Types in Map Algebra

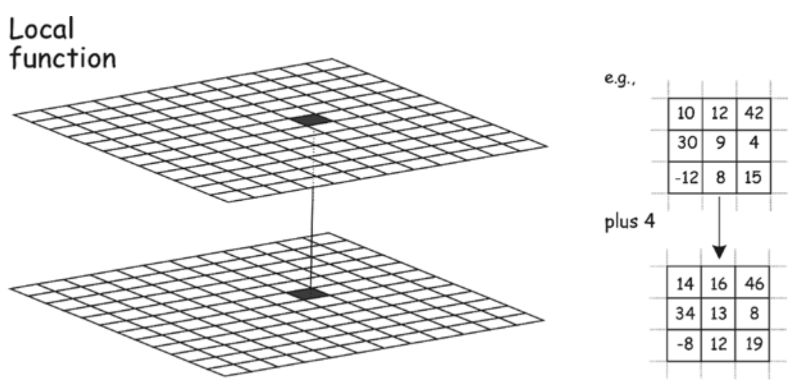
- **Local**: cell-by-cell
- **Neighborhood** (Focal): neighborhood analysis
- **Block**: whole blocks of cells
- **Zonal**: within homogeneous areas, zonal analysis/statistics
- **Global**: incorporation of the full dataset
- Special types

- **Uniform** definition of **entities** in raster data



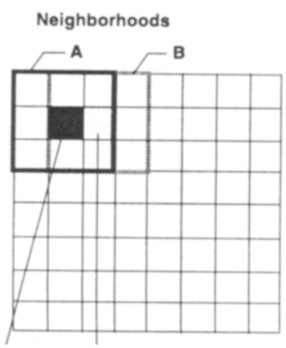
# Local Functions

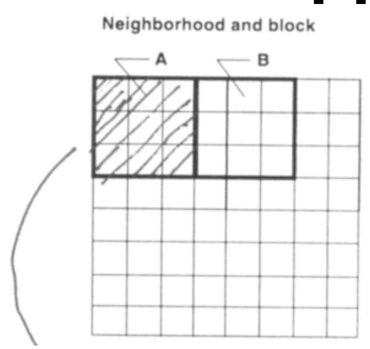
- **Uniform** cell size presumed for cell-by-cell analysis (**ul** start)
- **Mathematical/arithmetic** functions
- **Logical** operations
- **Reclassification**
- **Nested** functions
- **Overlay** in raster data
- `sin(c\:temp\myRaster)`

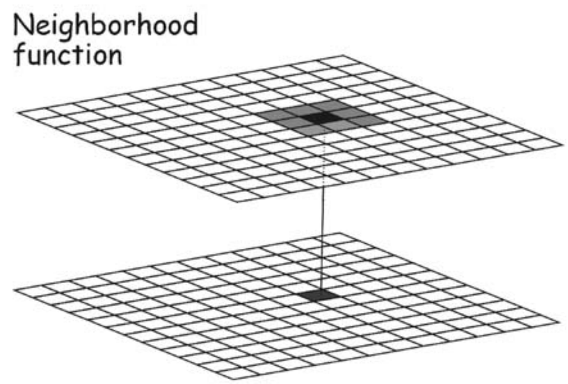


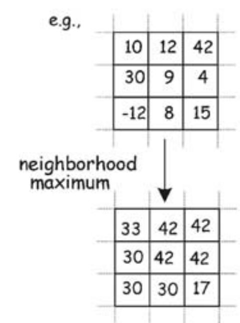
# Neighborhood (Focal) Functions

- Everyday needs in raster GIS (slope, aspect)
- Also neighborhoods **uniform**
- Cell is characterized/modified based on a predefined **neighborhood** of grid cells
- **`Focalsum([inlayer], rectangle, 3, 3)`**
- **Neighborhoods** and **moving windows**
- **Blocks non-overlapping**



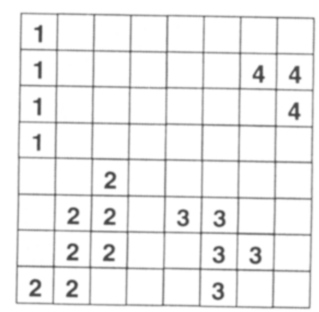



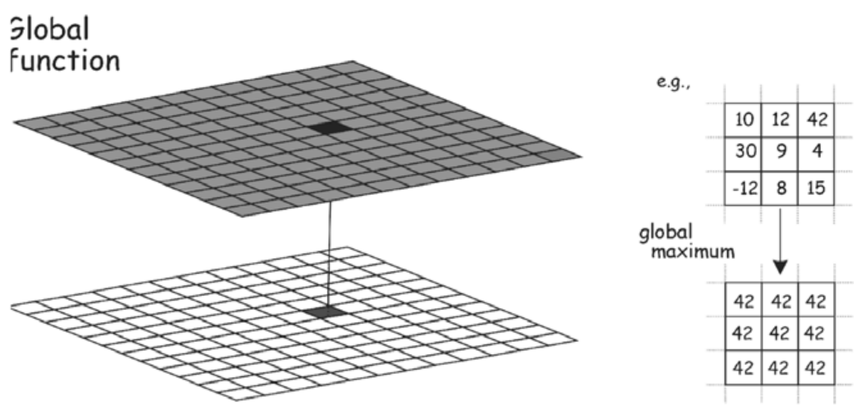
# Zonal Functions

- Zones identified from **another** layer for evaluation of the target cell
- Zones are **geographic areas** with certain characteristics (not necessarily connected such as regionGroups)
- Internal **attribute homogeneity**
- **`meangrid = zonalmean(zonalgrid,valuegrid)`**
- Statistics are written **into each cell of a zone**
- **Blocks** similar but predefined “roving window corresponds to the zone



# Global Functions

- Operate on the **entire grid** at once
- Each cell is a **function** of all input cells
- EucDistance, WeightDistance, Hydrological, Surface, Visibility, Viewshed, Max,…
- **`eucdist(…)`**



# Further Functions

- We will see many specialized functions implemented in GIS tools
- Complex functions which **integrate** many of **basic** functions
- **Hydrological** functions
- **Surface** analysis

# Flow Control

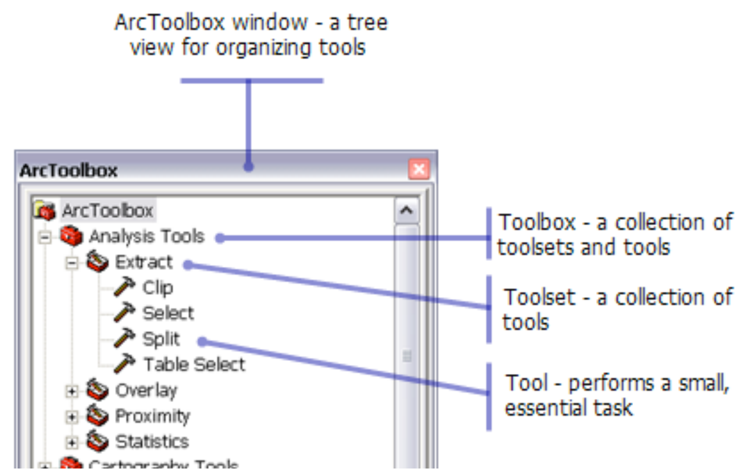
- Integral component of Map Algebra
- **Command line** framework / **GUI** as **interface** with the GIS user
- Composed of two elements (that work with **operations** and **functions)**:
- **Statements**: verbal representation of operations to link operators, functions and programming commands
- **Programs**: notational representation of a procedure in Map Algebra; ordered sequence of statements

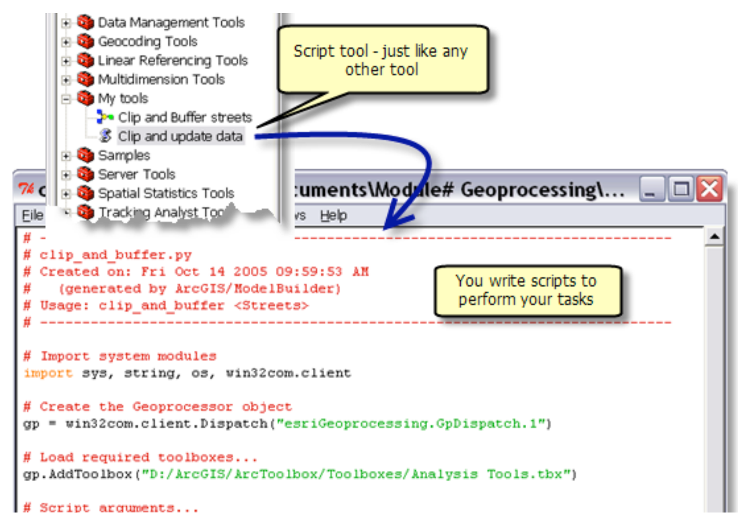
# Iteration

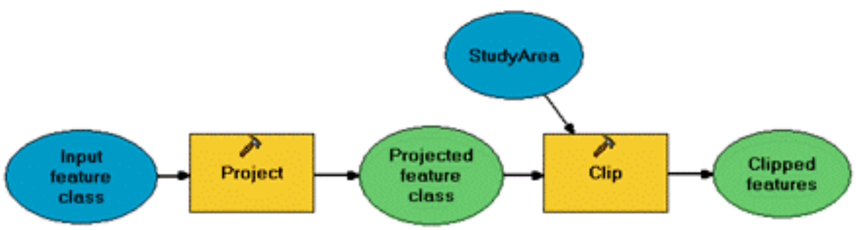
- **Repeated** execution of the same sequence of statements under **varying conditions** or with other **datasets** or subsets or based on **testing** for **conditions** (“for looping”, “while looping”)
- Combined with **if then** logic
- Makes a modeling framework more **powerful**

# ... Geoprocessing Framework in ArcGIS

- **Tool dialog**: Single tasks
- **Command line**
- **Model-building**: This is GIS2!
- **Scripting/Programming**: This is where we'll go in GIS3!

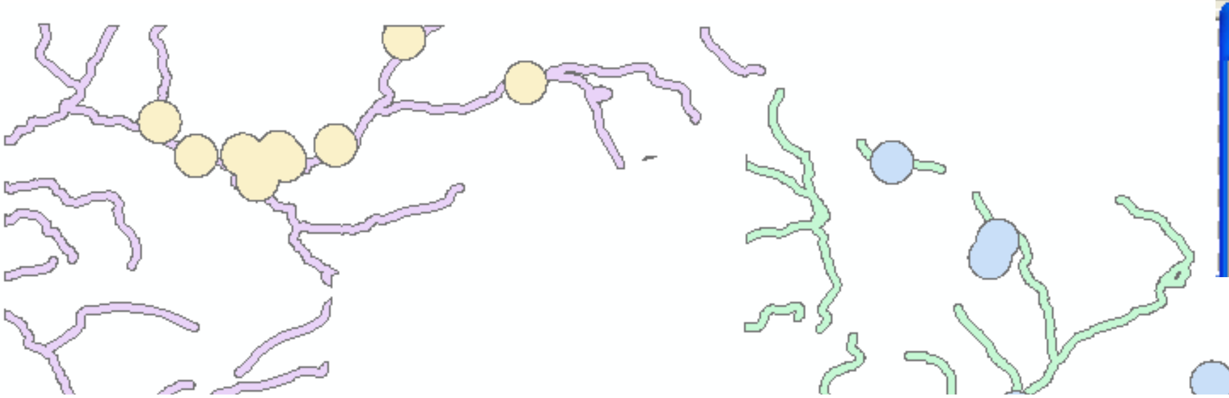


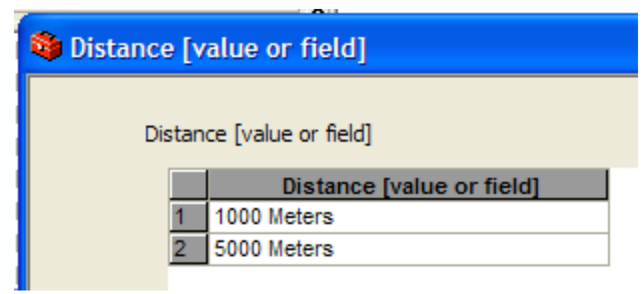



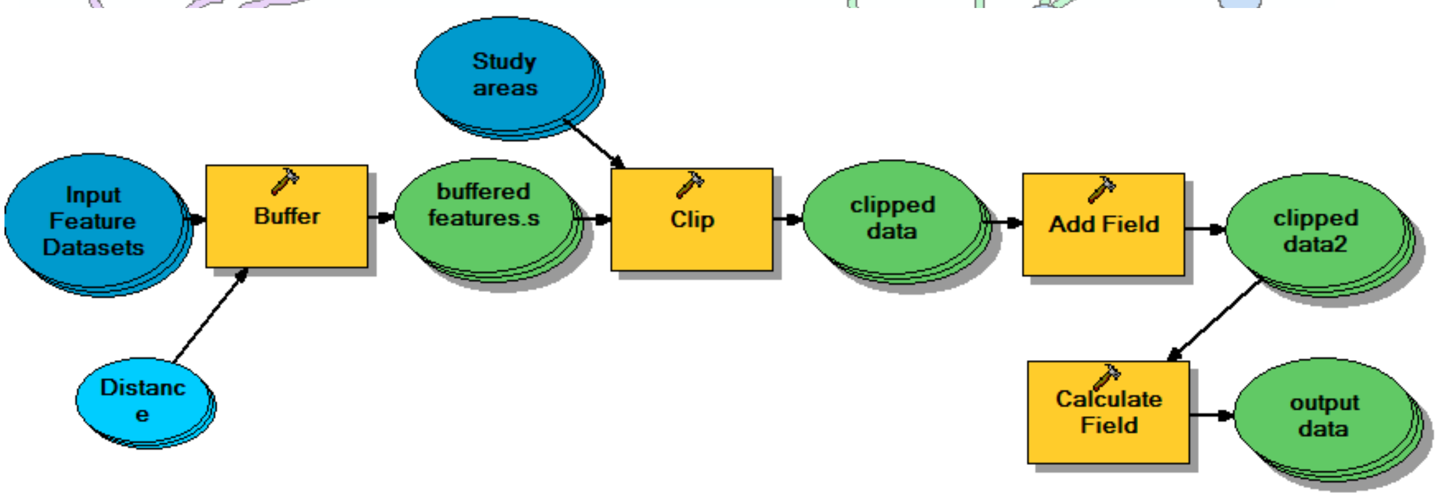



# “Graphical” Modeling Framework

Different Buffer distances for different features in different areas

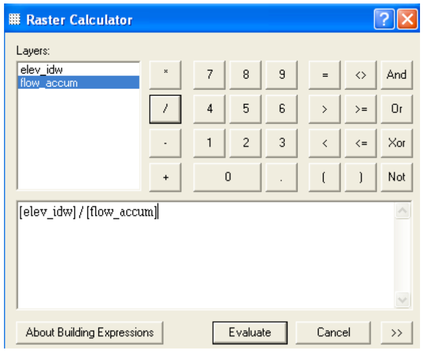


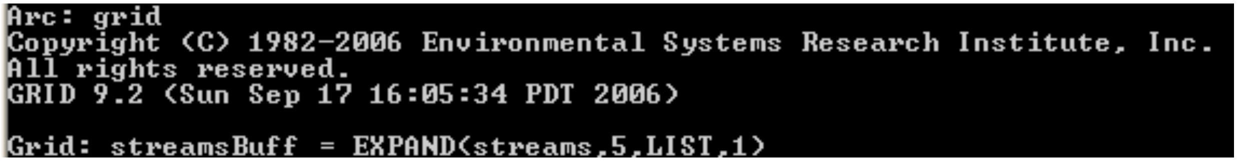


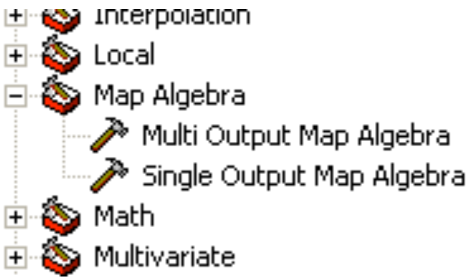



# How to Access Map Algebra

- Spatial Analyst: **Raster Calculator**
- Command line prompt in **ArcGrid** (going to be gone…)
- ArcToolbox: **Single Output Map Algebra** (for use in ModelBuilder)
- ArcToolbox: **Multiple Output Map Algebra** (for use in scripting)



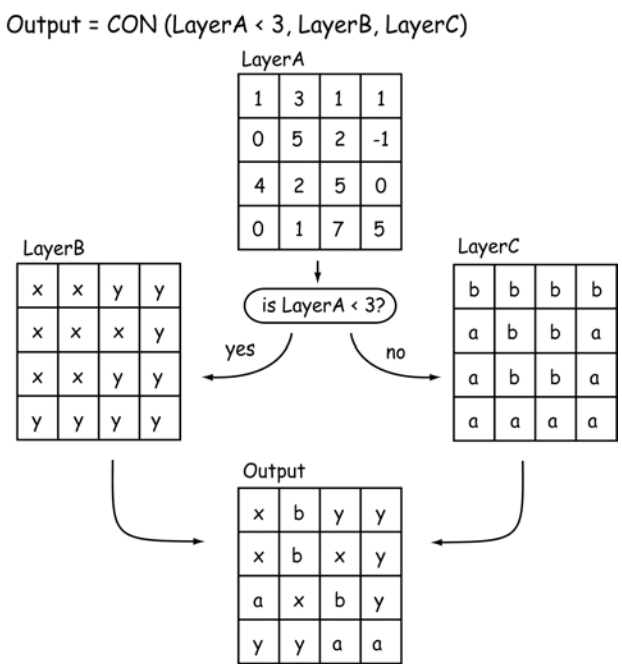



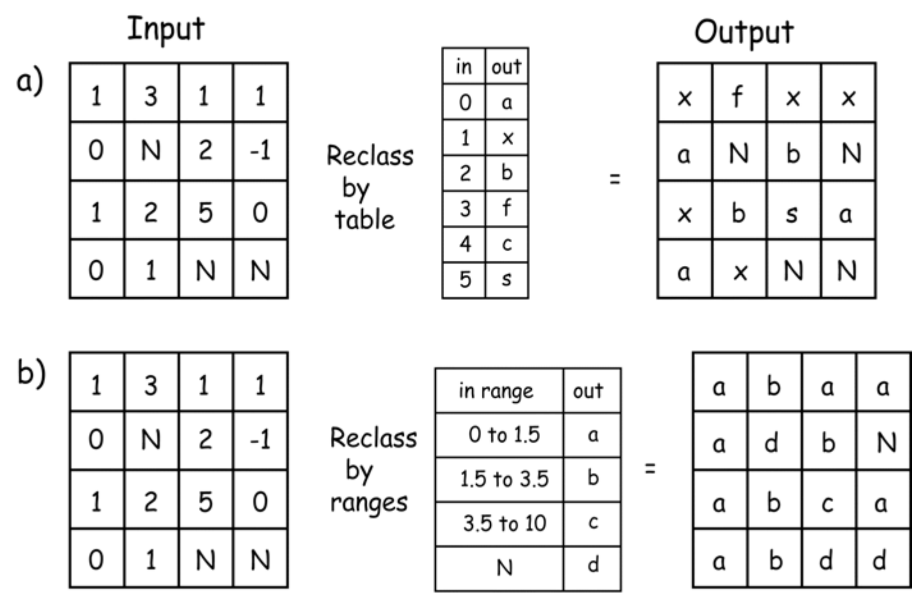
# More on Local Functions

- Basically the **operators** we have seen before are executed **cell-by-cell**
- Remember the handling with **NoData** values, **0-values** and values **unequal** to **zero**
- Higher level operations based on local processing are **reclassifications**, **nested functions** and **overlays**…
- …trigonometric, exp., log., select, statistical,…

# Local Functions: Reclassifications

- Assigning output values that depend on the specific set of input values
- Based on a **table**, **ranges of values** (for automated reclassification) or **conditional tests**

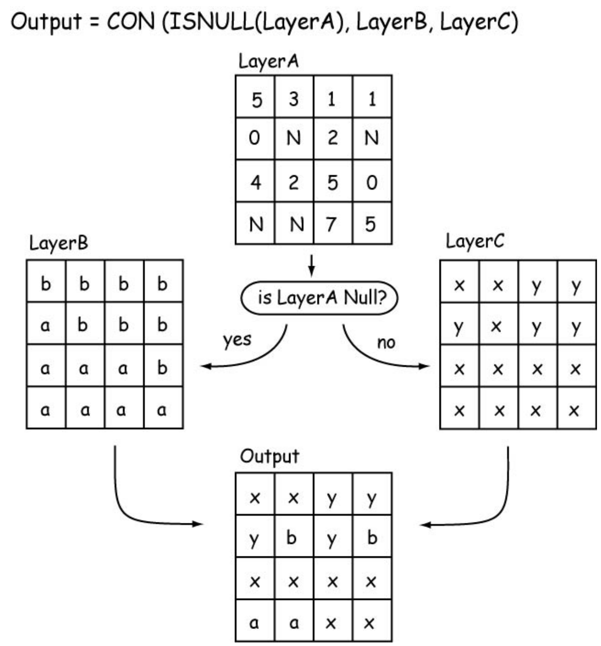


# Local Functions: Nested Functions

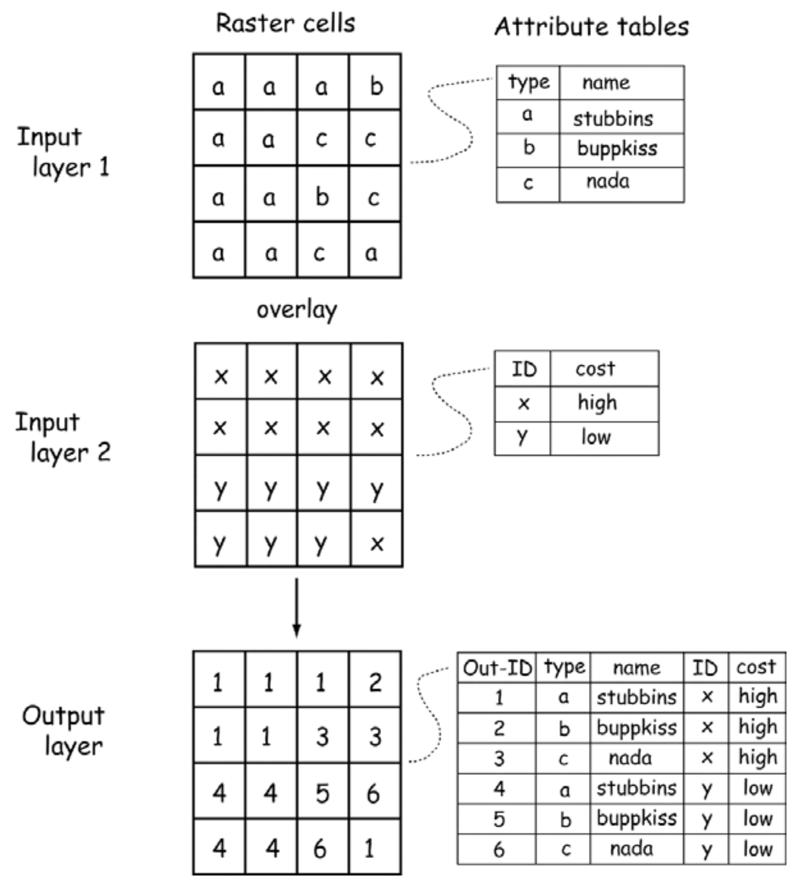
- Functions as arguments in other functions

```
absValues = ABS(Input)
LogValues = LN(absValues)
LogValues = LN(ABS(Input))
```



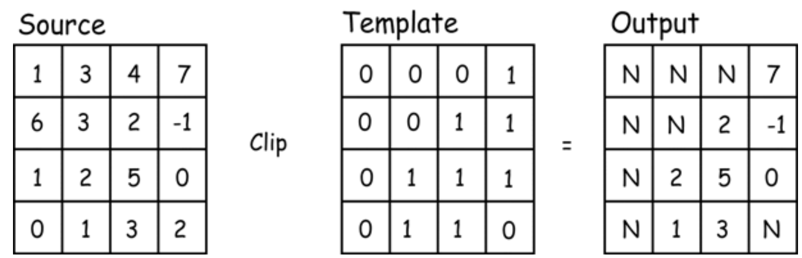
# Local Functions: Overlay I

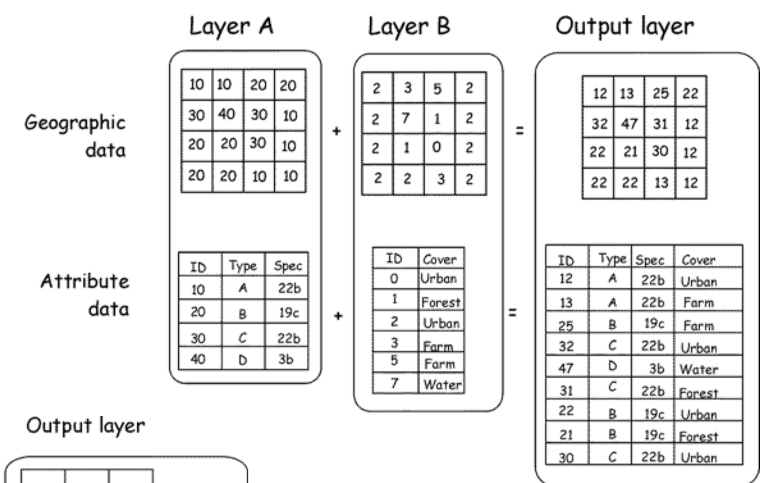
- **Cell-by-cell comparison** to register unique combinations of variables
- “**vectors**” of attributes given in a **table** (many to one relationships and extended rasters)



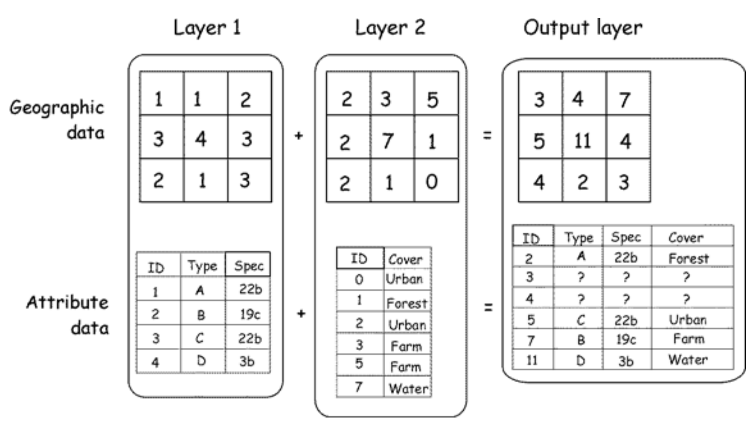
# Local Functions: Overlay II

- Multiplication to mimic **clip** (**extraction**) using source and template layers (binary masks)



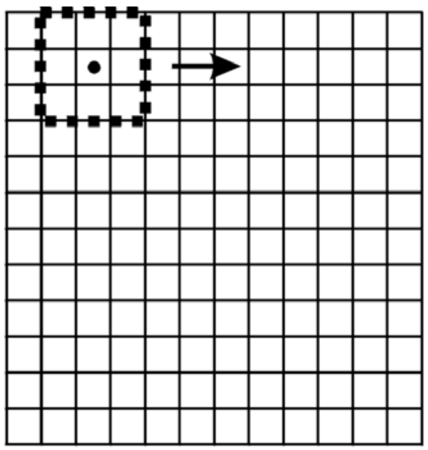

- **Addition** to mimic **union** (same id’s for disjunct pixels with same characteristics through many-to-one rs. + ambiguous if same values out of different combinations)




Combining zonal with overlay…

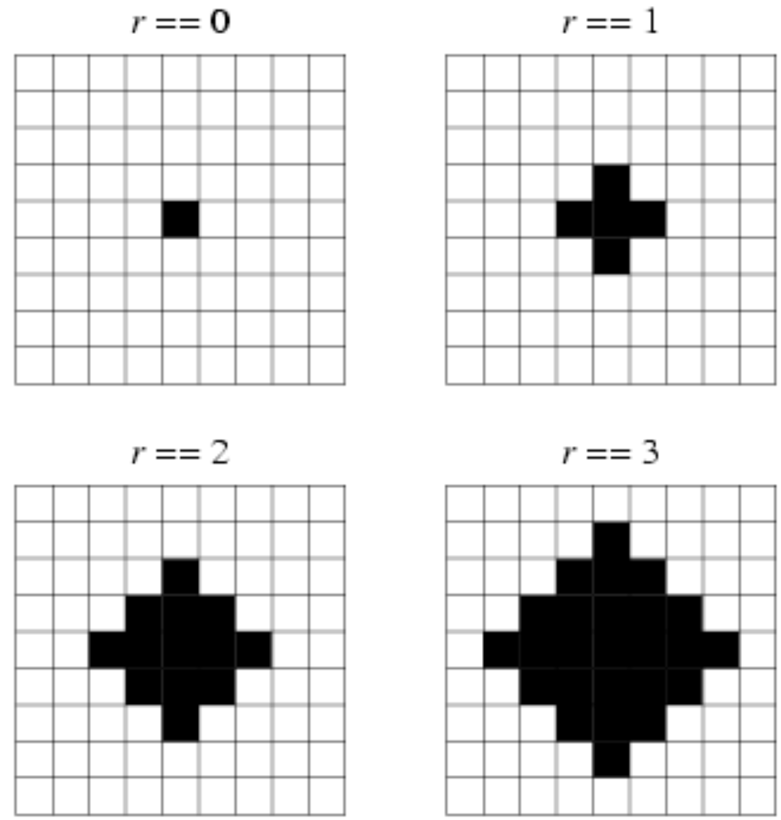
# More on Neighborhood Functions

- Often based on the concept of **moving windows**:
- Configuration of raster cells that is positioned over the input raster and defines the input for an **operation** to be applied. Result associated with **center** and written to the **output**. Window "**moves**" to the next location…
- Much depends on the **neighborhood**…
- Any ideas of functions that use NF?



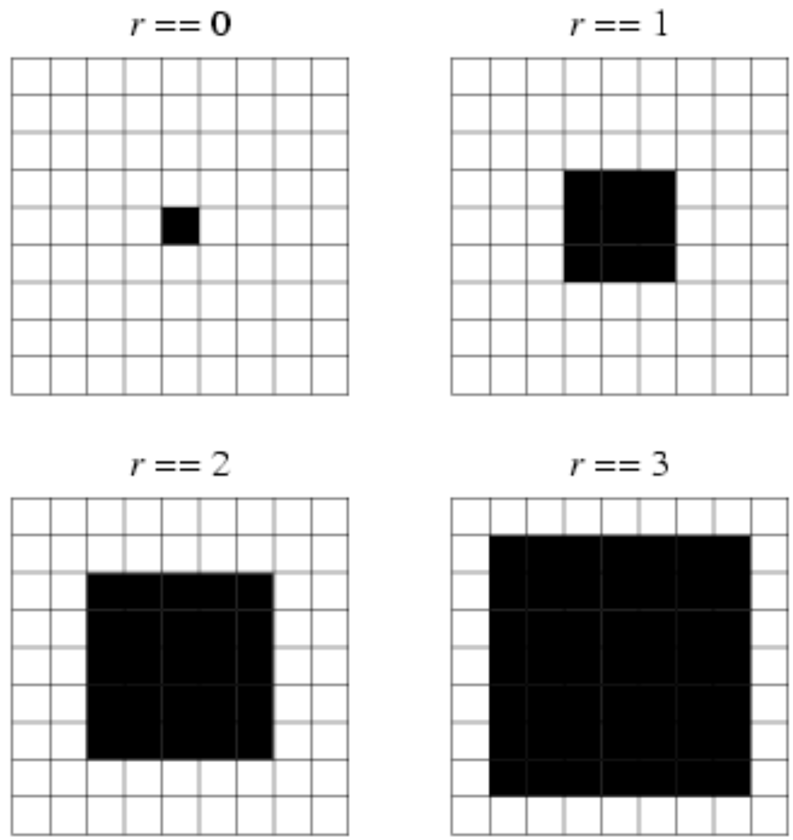
# Von Neumann Neighborhood

- **Diamond-shaped**
- To define a set of cells surrounding a given cell
- Ranges `r = 0,1,2,3`
- `N = 2*r(r+1)+1`
- "centered square number"

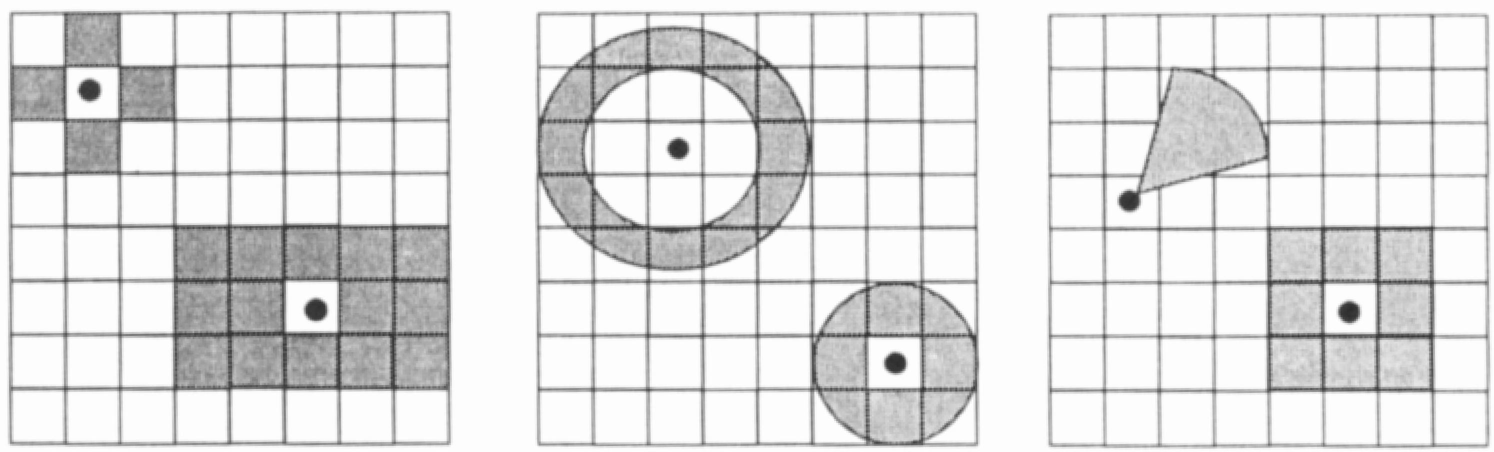


# Moore Neighborhood

- **Square-shaped**
- To define a set of cells surrounding a given cell
- Ranges `r = 0,1,2,3`
- $N = (2r+1)^2$
- "**odd** squares"

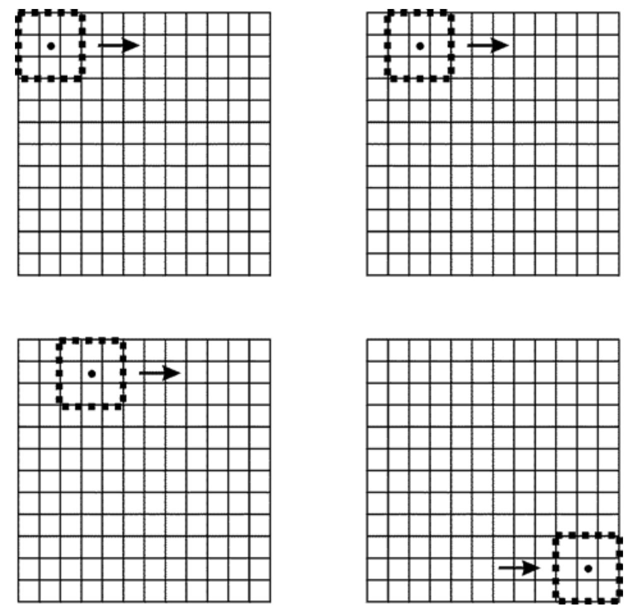


# Other Types of Neighborhood

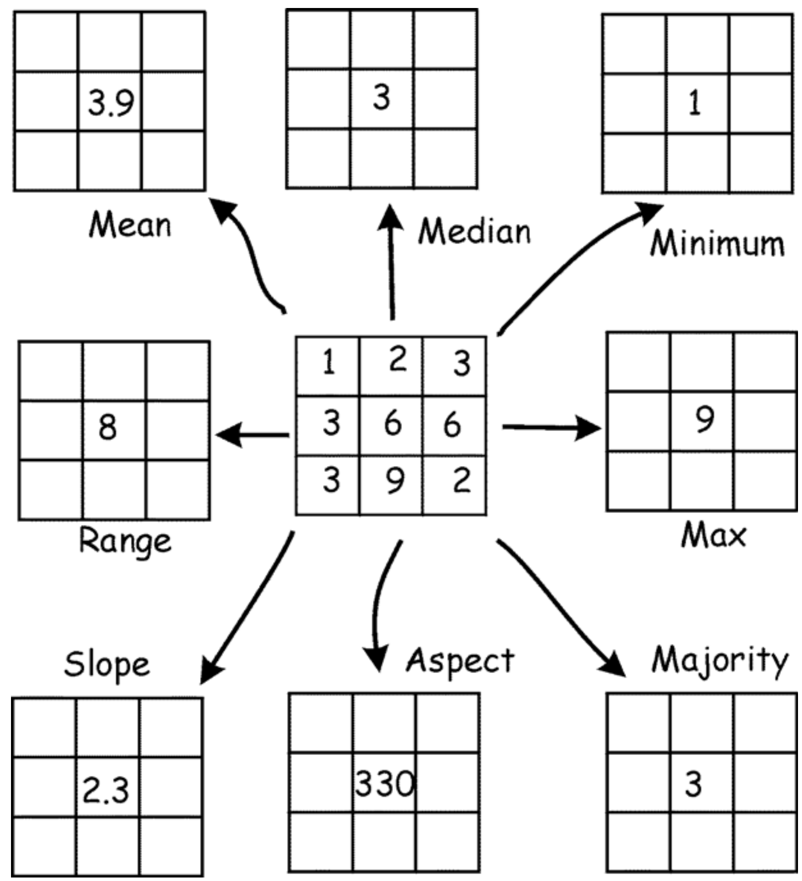


# The Principles of Moving Windows

- Left-to-right and top-to-bottom
- **Dimensions**: size of the neighborhood
- **Odd-numbered** in x and y to provide a **natural center** cell and square- or rectangular-shaped (or L or wedge or circular)

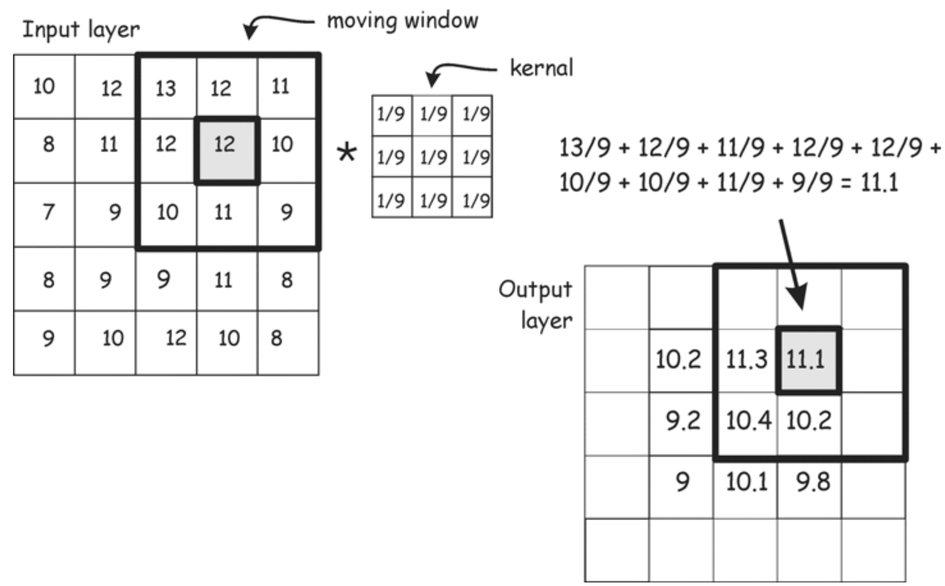


# Examples for Moving Window NFs



# Moving Windows and Kernels

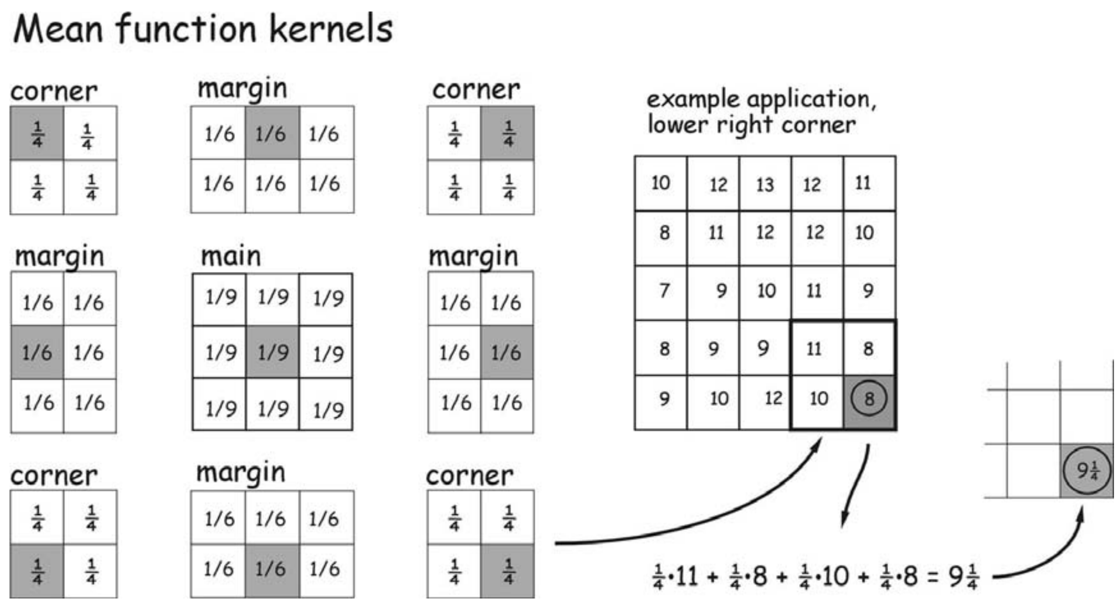
- Set of **constants** for a given window size and shape
- Are applied with a **function** to every moving window **location**
- What can you see at the **margins** of the output grid?



# Moving Windows and Margin Erosion

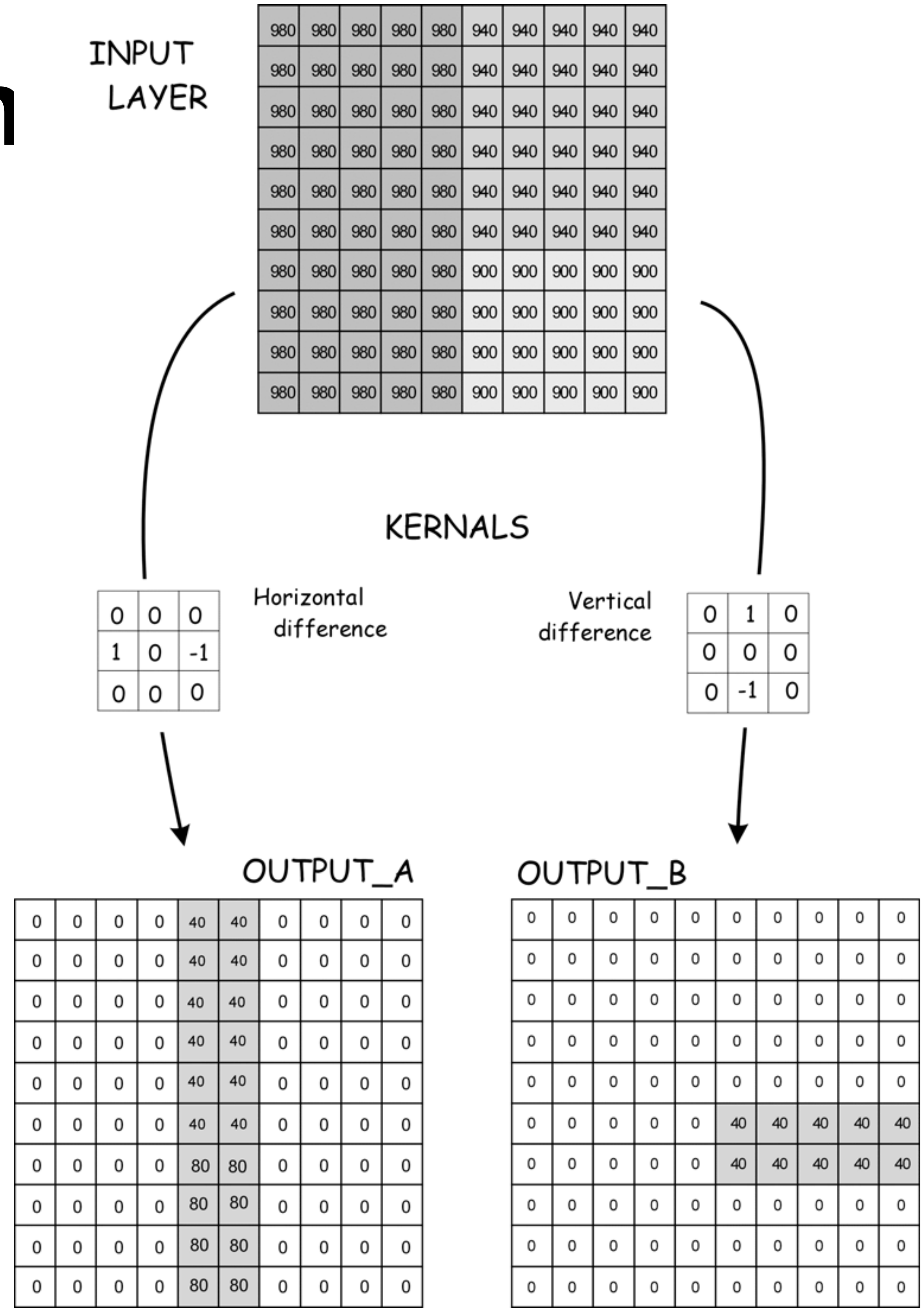
- Loss of margins in the original raster dataset: width of cells from the center cell away
- Solutions: (a) **Enlargement** of study areas

  (b) **MW-** and **Kernel modification** at corners (2x2; 1/4) and edges (2x3; 1/6)



# Edge Detection Using Kernels

- Concentration changes; elevation changes,…
- Discovering **contrasts / differences** within the local neighborhood



# Noise Filtering Using Kernels

- **Smoothing**: Reducing differences between neighboring cells (mean Kernels / operators; **low-pass filters**)
- **Noise**: Errors in measurement, recording or transforming the data or due to data loss

# High-Pass Filter

- Accentuates differences between adjacent cells in a moving window
- Identifying “spikes” (+ >>) and “pits” (- >>)

kernal for high-pass filter

$$\frac{1}{9} * \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 9 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$\left( \frac{\begin{array}{l} (-1)*1065 + \\ (-1)*1068 + \\ (-1)*924 + \\ (-1)*1038 + \\ (\ 9)*963) + \\ (-1)*947 + \\ (-1)*1142 + \\ (-1)*1005 + \\ (-1)*1151 \end{array}}{9} \right) = 35.7$$

Input Layer with “noise”

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1065 | 1068 | 929 | 864 | 960 | 1113 | 974 | 896 | 890 | 841 | 759 | 719 | 705 | 696 | 720 | 708 |
| 1038 | 963 | 947 | 950 | 999 | 1021 | 1011 | 1015 | 995 | 1044 | 870 | 773 | 734 | 703 | 676 | 684 |
| 1142 | 1005 | 1151 | 310 | 1117 | 1056 | 1007 | 1002 | 902 | 954 | 935 | 913 | 789 | 756 | 724 | 700 |
| 1116 | 1114 | 1270 | 1165 | 1097 | 1025 | 922 | 917 | 821 | 829 | 860 | 838 | 807 | 810 | 758 | 760 |
| 1275 | 1170 | 1295 | 1114 | 1009 | 942 | 953 | 847 | 835 | 729 | 738 | 797 | 723 | 718 | 694 | 670 |
| 1441 | 1263 | 1196 | 1055 | 913 | 869 | 829 | 771 | 736 | 765 | 766 | 688 | 694 | 1676 | 684 | 698 |
| 1348 | 2900 | 1056 | 969 | 948 | 951 | 940 | 867 | 818 | 863 | 784 | 732 | 704 | 733 | 776 | 804 |
| 1377 | 1238 | 1122 | 1019 | 1089 | 950 | 956 | 896 | 2000 | 800 | 760 | 698 | 779 | 867 | 896 | 744 |
| 1489 | 1320 | 1188 | 1152 | 1050 | 942 | 922 | 952 | 815 | 841 | 721 | 780 | 852 | 28 | 845 | 738 |
| 1432 | 1415 | 1196 | 1100 | 1001 | 974 | 924 | 911 | 914 | 756 | 809 | 861 | 898 | 830 | 746 | 710 |
| 1412 | 1474 | 1240 | 1100 | 1001 | 982 | 873 | 835 | 829 | 853 | 931 | 937 | 845 | 706 | 685 | 680 |
| 1493 | 1368 | 1201 | 1090 | 1064 | 970 | 902 | 902 | 958 | 952 | 1015 | 841 | 782 | 803 | 786 | 711 |
| 1437 | 1407 | 118 | 1145 | 1070 | 1107 | 982 | 1047 | 1077 | 1052 | 954 | 884 | 44 | 940 | 828 | 771 |
| 1349 | 1369 | 1267 | 1247 | 1194 | 1196 | 1077 | 1214 | 1145 | 999 | 906 | 894 | 1024 | 1046 | 923 | 862 |
| 1319 | 1292 | 1378 | 1400 | 1367 | 1276 | 1162 | 1088 | 961 | 930 | 872 | 985 | 1010 | 1178 | 1148 | 1000 |

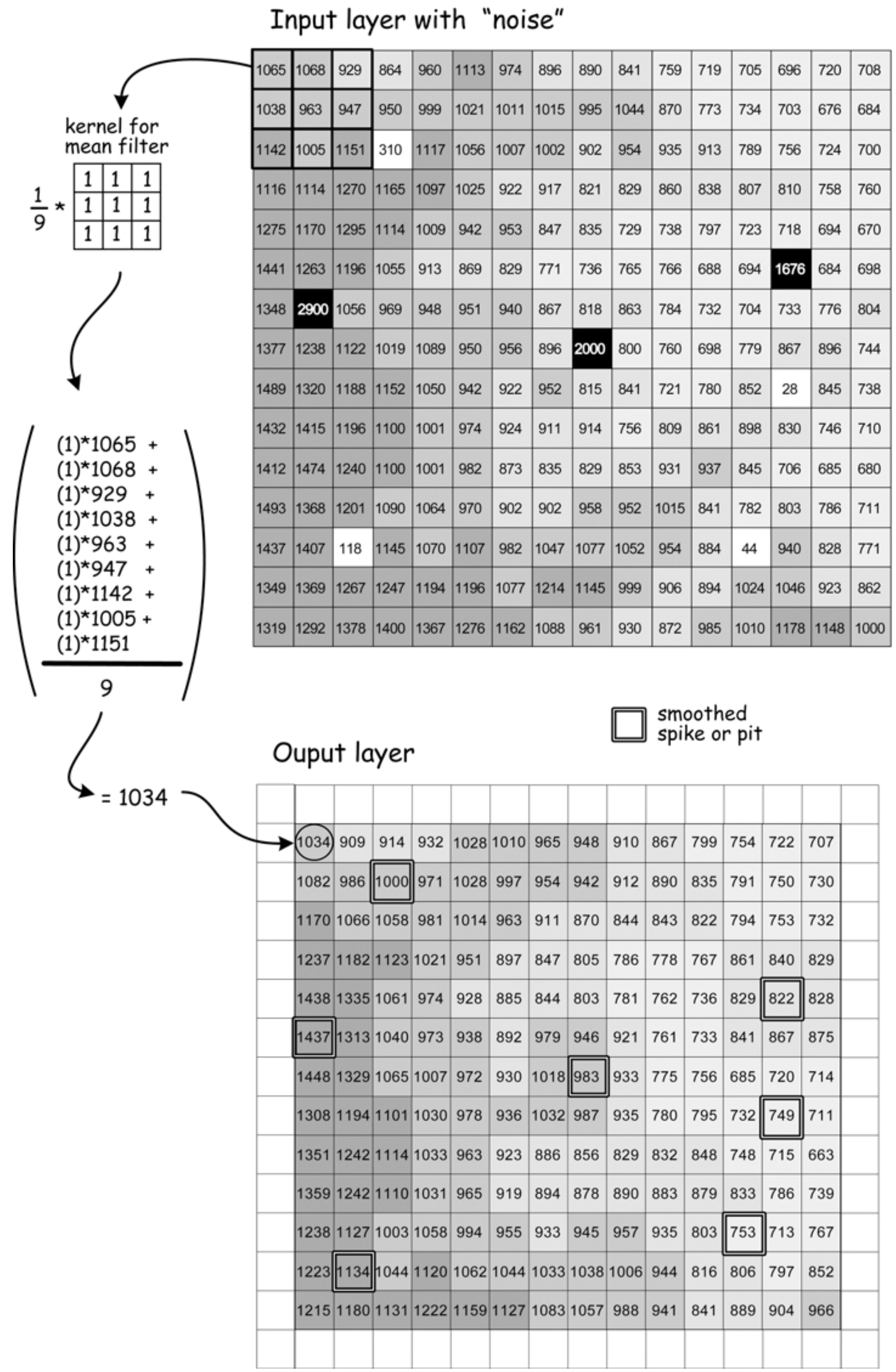
spikes ■ pits □

Ouput Layer

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 35 | 142 | 141 | 177 | 105 | 112 | 162 | 156 | 250 | 99 | 59 | 61 | 58 | 43 | |
| | 33 | 292 | -657 | 270 | 145 | 121 | 158 | 60 | 147 | 148 | 179 | 85 | 89 | 74 | |
| | 66 | 345 | 235 | 237 | 124 | 61 | 107 | 41 | 76 | 111 | 108 | 102 | 146 | 110 | |
| | 62 | 256 | 114 | 100 | 95 | 161 | 93 | 122 | 23 | 41 | 117 | -58 | -43 | -59 | |
| | -35 | -7 | 110 | 40 | 37 | 35 | 12 | 14 | 68 | 88 | 28 | -59 | 1039 | -69 | |
| | 1784 | -140 | 35 | 79 | 118 | 152 | -16 | -38 | 37 | 109 | 79 | -59 | -54 | -14 | |
| | -74 | -83 | 66 | 202 | 83 | 131 | -23 | 1238 | -45 | 69 | 18 | 179 | 243 | 281 | |
| | 158 | 125 | 178 | 135 | 68 | 88 | 25 | -82 | -1 | 20 | 71 | 214 | -718 | 227 | |
| | 220 | 86 | 108 | 78 | 119 | 102 | 126 | 159 | 10 | 66 | 108 | 249 | 207 | 165 | |
| | 278 | 135 | 111 | 80 | 125 | 50 | 33 | 42 | 57 | 150 | 161 | 105 | -3 | 21 | |
| | 281 | 207 | 207 | 123 | 83 | 46 | 68 | 119 | 99 | 192 | 130 | 115 | 179 | 105 | |
| | 340 | -1004 | 228 | 68 | 167 | 47 | 129 | 158 | 162 | 115 | 166 | -758 | 247 | 67 | |
| | 306 | 227 | 253 | 104 | 169 | 69 | 265 | 215 | 121 | 64 | 151 | 248 | 257 | 59 | |
| | | | | | | | | | | | | | | | |

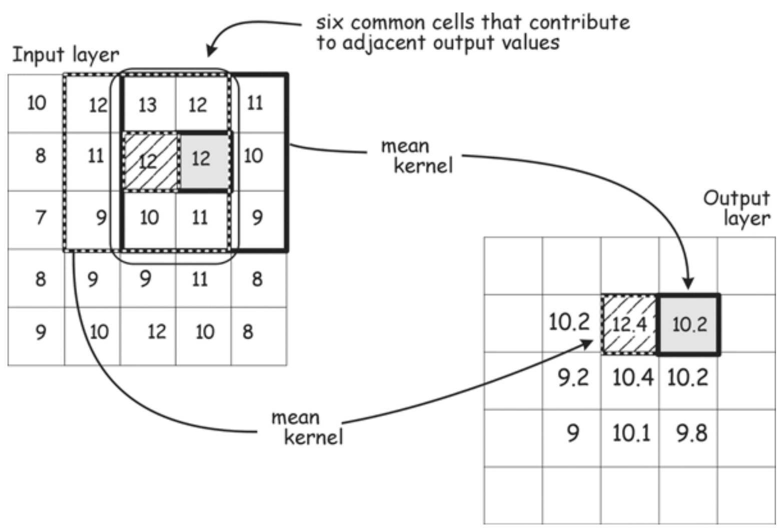
# Mean Filtering

- Remove the spikes and pits identified by high-pass filtering



# Moving Windows and Spatial Covariance

- The more moving window operations / functions carried out the more related (autocorrelated) the cell values are
- Adjacent cells share six of nine cells in the local neighborhood for the computation…



# References


- Longley P.A., M. F. Goodchild, D. J. Maguire and D. W. Rhind. 2005. Geographic Information Systems and Science. Second Edition. John Wiley, Chichester, 2005.
- Burrough, P.A. and McDonnell, R.A. 1998. Principles of Geographical Information Systems. London: Oxford.
- Tomlin, C.D. 1991 Cartographic Modeling. In Maguire, D., Goodchild, M.F., and Rhind, D. (Eds.) Geographic Information Systems: Principles and Applications. London: Longman: 361 - 374.
- Tomlin, C.D. 1983. An introduction to the Map analysis package. Proceedings. mNational Conference on Resource Management Applications: Energy and Environment, San Francisco. Pp. 1-14.