Geographic Information Technology Training Alliance (GITTA) presents:

Structures for Data Compression

Responsible persons: Claudia Dolci, Dante Salvini, Michael Schrattner, Robert Weibel

Content

1. Structures for Data Compression 2
1.1. General Compression Concepts 3
1.1.1. Data Storage 4
1.1.2. Compression Concepts 6
1.2. Vector Data Storage and Compression 10
1.2.1. Vector Data Model and Storage 10
1.2.2. Vector Data Compression 11
1.3. Raster Data Storage and Compression 14
1.3.1. Raster Data Structure 14
1.3.2. Raster Data Storage 16
1.3.3. Raster Data Compression 16
1.4. Image Formats 24
1.4.1. Digital Image Formats 24
1.5. Summary 29
1.6. Bibliography 30

1. Structures for Data Compression

The goal of data compression is to represent an information source as accurately as possible using the smallest storage space (=number of bits).
This lesson is focused on the compression concepts and techniques. After an introduction of some of the most important general compression concepts, the lesson makes a distinction between different types of data. The closer examination of various techniques of data compression allows you to become familiar with this topic.

Learning Objectives

- You know the meaning and the basic principles of data storage and compression concepts.
- You are able to explain the structure and difference of vector and raster data, in particular the storage and compression of data. Furthermore you are able to distinguish between lossless and lossy data compression and know their specific characteristics.
- You know the most widely used image formats and be able to describe their properties as well as their advantages and disadvantages.

1.1. General Compression Concepts

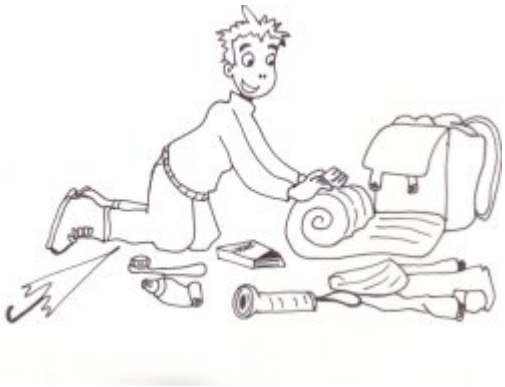
Why do we need compression?

Have you ever tried to pack your car for the summer holiday?
Organizing the space in the car could be compared to the data compression problem. Large amounts of data can create enormous problems in **storage space** and **transmission time**.
The main reasons of data compression could be summarised as:

- Multimedia data have large data volume
- Difficulty sending real-time uncompressed data over current network

The widespread consumer-market use of information in the form of images has contributed much to the development of data compression techniques. The design goal of image compression is to represent images with as few bits as possible, according to some **fidelity criterion,** to save storage and transmission channel capacity.

Compression data principle
"Eliminate data redundancy and try to find a code with less data volume."



All image compression techniques try to get rid of the inherent redundancy, which may be spatial (neighboring similarity or equal pixels), spectral (pixels in different spectral bands in a color image) or temporal (correlated images in a sequence, e.g. television).

1.1.1. Data Storage

How do we measure the data storage in a computer?
A computer stores information in **binary format**. The Binary system is a number system which uses **bits** to store data. A **bit** is a *"binary digit"*, the smallest increment of data on a machine. A bit can hold only one of two values: 0 or 1.
Because bits are so small, you rarely work with information one bit at a time. Bits are usually assembled into a group of 8 to form a **byte** or *"binary term"*. Computer memory is typically byte addressed – each byte has a unique address.
Bytes are often used to store characters (they contain enough information to store a single character), but they can also be used to store numerical values.
A byte can store a numerical value between 0 and 255 or between -127 and 127 if we are considering the negative numbers too.
There are $2^8 = 256$ different byte values:

```
11111111
11111110
...
00000011
00000010
00000001
00000000
```

Bit values

For the purposes of storing numerical data values, bytes are grouped together into **words,** which are typically 2 bytes.

short int, unsigned short	16 bits (2 bytes)
int, unsigned int	32 bits (4 bytes)
long int, unsigned long	32 or 64 bits (4 or 8 bytes)

Data type definition

Data units of 512 bytes or more are called **data blocks**. Each operating system has a specific block size.

1 byte	8bits	
1 kilobyte	2^{10} bytes	1’024 bytes
1 megabyte	2^{20} bytes	1’048’576 bytes
1 gigabyte	2^{30} bytes	1’073’741’824 bytes
1 tetrabyte	2^{40} bytes	1’099’511’627’776 bytes

Conversion table

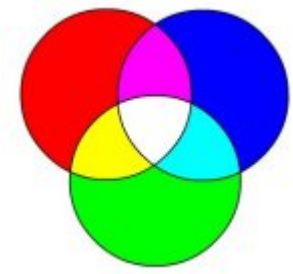
Example
A good example is given by **digitized images**: a single DIN A4 (8.2 x 11.6 inches) color picture (4 colors) scanned at 300 dpi with 8 bits/pixel/color, produces 30 MBytes of data.
Challenge: Try to calculate how to arrive at that result (30MB)

Solution

8.2 x 300 dpi = 2’460
11.6 x 300 dpi = 3’480
2’460 x 3’480 = 8’560’800
8’560'800 x 4 (colors) = 34'243’200 bytes approx. 30 MBytes

Storing characters
“A”= 01000001
“B”= 01000010
“C”= 01000011

Storing colors
The computer screen uses a RGB (Red/Green/Blue) model, in which each color is represented by 16 bits.



In the following table the 16 principal Windows colors are illustrated.

	Color	RGB Binary Code
	BLACK	000000000000000000000000
	WHITE	111111111111111111111111
	RED	111111110000000000000000
	LIME	000000001111111100000000
	BLU	000000000000000011111111
	YELLOW	111111111111111100000000
	CYAN	000000001111111111111111
	MAGENTA	111111110000000011111111
	MAROON	100000000000000000000000
	GREEN	000000001000000000000000
	NAVY	000000000000000010000000
	OLIVE	100000001000000000000000
	TEAL	000000001000000010000000
	PURPLE	100000000000000010000000
	SILVER	110000001100000011000000
	GRAY	100000001000000010000000

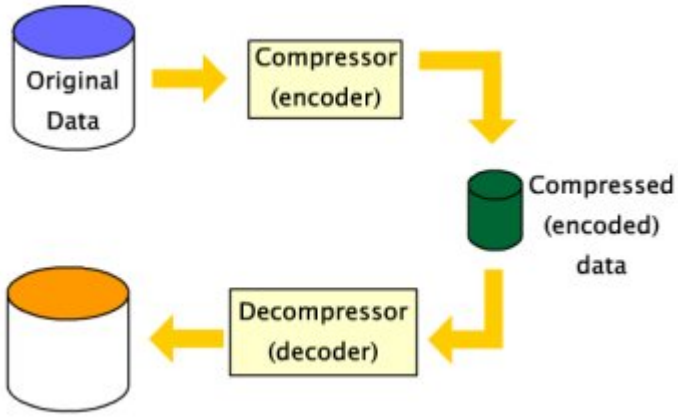
1.1.2. Compression Concepts

Data compression principles

Below, data compression principles are listed. Data compression:

- Is the substitution of frequently occurring data items, or symbols, with short codes that require fewer bits of storage than the original symbol.
- Saves space, but requires time to save and extract.
- Success varies with type of data.
- Works best on data with low spatial variability and limited possible values.
- Works poorly with high spatial variability data or continuous surfaces.
- Exploits inherent redundancy and irrelevancy by transforming a data file into a smaller one.

The **compression ratio** is the ratio of the two file sizes. e.g., original image is 100MB, after compression, the new file is 10MB. Then the compression ratio is 10:1.



Two compression techniques are mostly used: **Lossless** and **Lossy** compression.

Lossless compression

A **lossless compression** algorithm eliminates only redundant information, so that one can recover the data exactly upon decompression of the file.

Lossless data compression is compression without any loss of data quality. The decompressed file is an exact replica of the original one. Lossless compression is used when it is important that the original and the decompressed data be identical. It is done by re-writing the data in a more space efficient way, removing all kinds of repetitions (compression ratio 2:1).

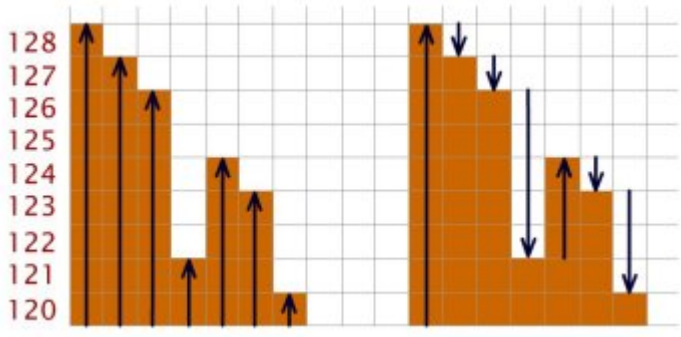
Some image file formats, notably PNG, use only lossless compression, while those like TIFF may use either lossless or lossy methods.

Examples of LOSSLESS METHODS are:

- run-length coding
- Huffman coding

- Lempel-Ziv-Welsh (LZW) method

Numerical example
An example of seven gray pixels:
128, 127, 126, 121, 124, 123, 120
Can be re-written in shorter numbers requiring less bits like:
128, -1, -1, -5, +3, -1, -3

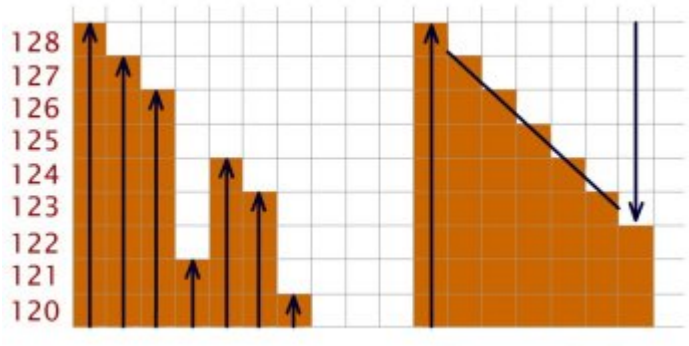


Lossless method

Lossy compression
A **lossy compression** method is one where compressing data and then decompressing it retrieves data that may well be different from the original, but is "close enough" to be useful in some way.
The algorithm eliminates irrelevant information as well, and permits only an approximate reconstruction of the original file. Lossy compression is also done by re-writing the data in a more space efficient way, but more than that: less important details of the image are manipulated or even removed so that higher compression rates are achieved. Lossy compression is dangerously attractive because it can provide compression ratios of 100:1 to 200:1, depending on the type of information being compressed. But the cost is loss of data.
The advantage of lossy methods over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any known lossless method, while still meeting the requirements of the application.
Examples of LOSSY METHODS are:

- PCM
- JPEG
- MPEG

Numerical example
The previous sequence of numbers
128, 127, 126, 121, 124, 123, 120
can be re-written like:
128 - 6
Result after decompression:
128, 127, 126, 125, 124, 123, 122



Lossy method

1.2. Vector Data Storage and Compression

Introduction

In this unit, vector data storage and the most often used data compression methods are presented.

1.2.1. Vector Data Model and Storage

Vector data model

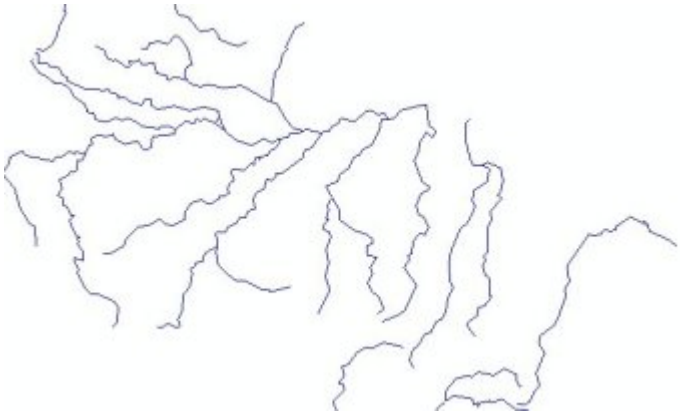
In a vector data model, the computer stores **points, lines** and **polygons** as x and y locations.
A point feature would only have one x and y location associated with it, whereas a line feature would be stored as a series of several x,y pairs. Polygons are also stored as a series of x and y locations, but the first and last position is the same location.

Vector data storage

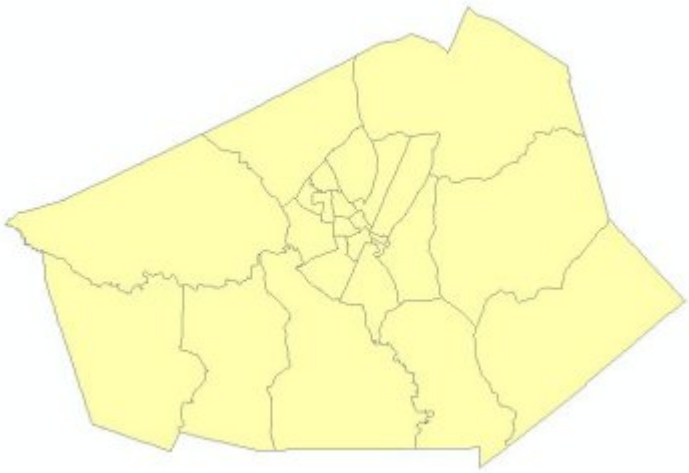
In today's systems, this information is often stored internally in relational tables which are organized according to a set of standard rules. The users do not normally see it. We do, however, see attribute information about the features. Features can have many different types of attributes.

Points

Lines



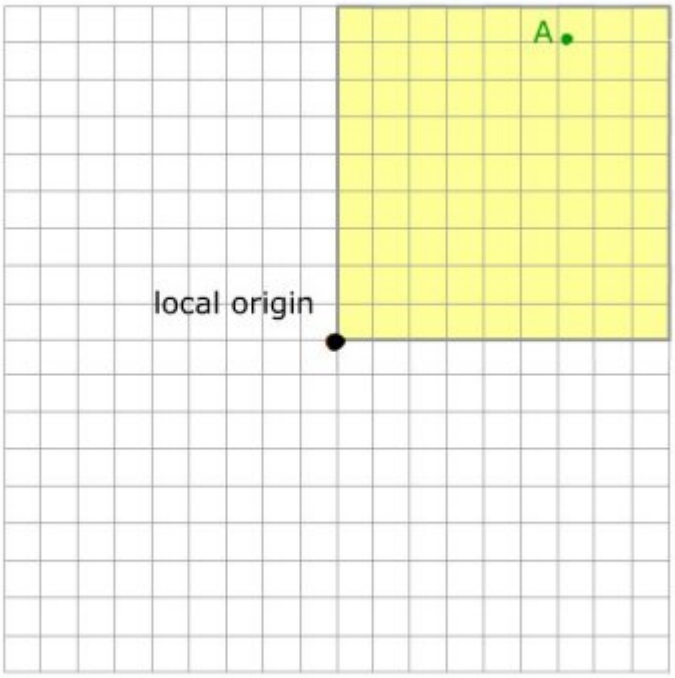
Polygons



1.2.2. Vector Data Compression

The spatial vector data can be compressed to save computer storage space.

Point data compression
If we are considering the UTM coordinates (Universal Transverse Mercator), points are described by an easting and northing coordinate, respectively.
For example:
Point A (897'345.32; 1'898'765.98)
If we are dealing with a relatively small area, we can imagine that coordinate values are stored with considerable redundancy, because for example the first two or three digits of all eastings and northings have the same values throughout the file.
To save storage space, it is possible to define a **LOCAL ORIGIN** and store all the coordinates relative to this new origin. These new coordinates will be smaller numbers than the original ones and can be stored in a smaller amount of computer storage space.
The coordinate offsets will preserve the **ABSOLUTE COORDINATES,** which will be used to restore the information when we use the data.



800,000; 1,800,000

Linear features compression

In this paragraph, we consider both linear features and polygons since a polygon is a sequence of lines.

The procedure described for the point features can also be applied to linear features where only the first point is saved with the full coordinate information. All remaining coordinates of each line use the local origin as reference. In this case, the local coordinates can be stored as short integer, saving 2 bytes per coordinates.

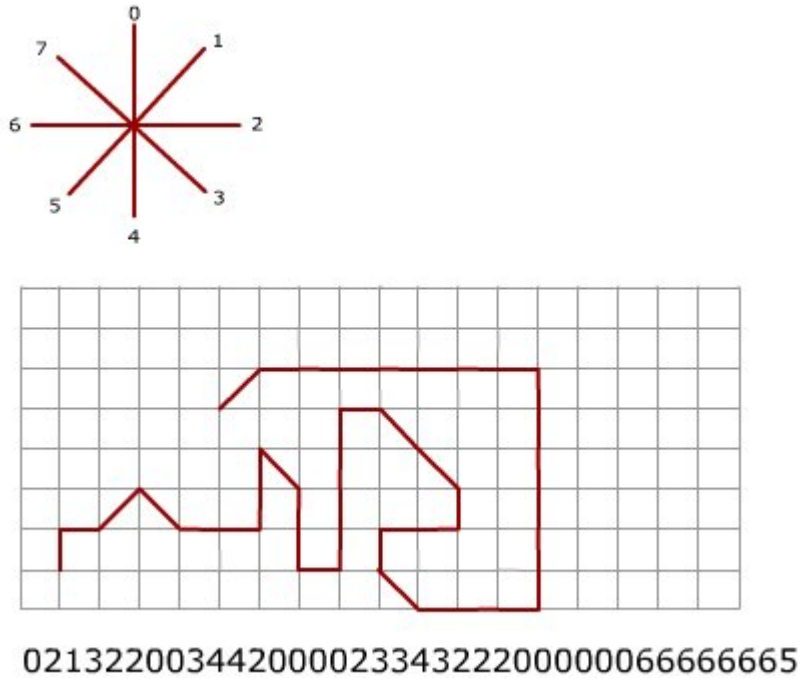
WARNING

With this compression we risk the lose of geometric precision.

Rasterising vector data and the Freeman coding

Another possibility suitable for lines is first rasterising the features onto a regular grid and afterwards proceeding with the Freeman coding or chain coding method. With this compression method, only the coordinates of the starting point of each line are stored.

All other subsequent points (pixels) are described by a number from 0 to 7 representing 8 possible directional positions in respect to the previous pixel. Three bits of storage would therefore be required for each pixel of the chain code.



Freeman coding with 8 directions

Increasing the number of directional positions from 8 to e.g. 16 (corresponding to 4 bits) or 32 (corresponding to 5 bits) it is possible to enhance the geometric precision.

CHARACTERISTICS

- Codes can be stored using integer types.
- Only boundary information is stored.
- Shape analysis is possible (perimeter, directional analysis, shape turns).
- Used in raster-to-vector conversion.
- All boundaries between regions are stored twice

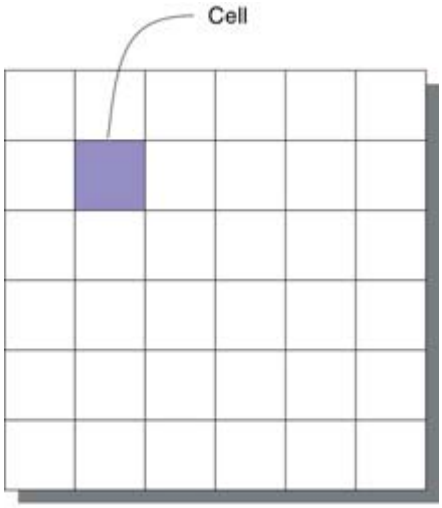
1.3. Raster Data Storage and Compression

Introducion

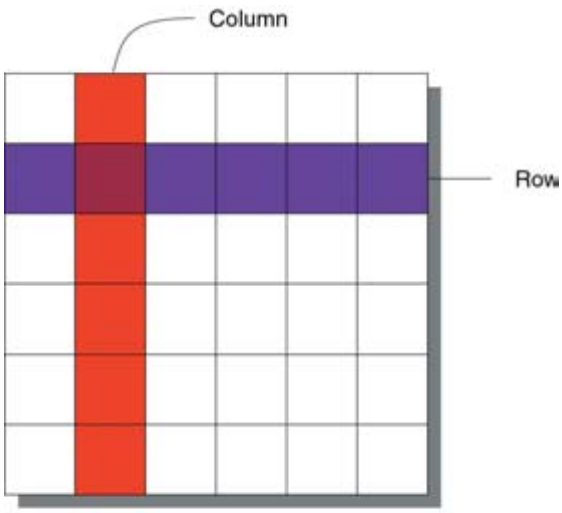
Raster data are becoming more important and widely used in geo-information science, particularly by Internet applications. In this unit, diverse examples and applications of raster data will be presented.
After an introduction and definition of raster data structure, raster data storage and compression methods are illustrated.

1.3.1. Raster Data Structure

Raster data model
A raster is an array of cells, where each cell has a value representing a specific portion of an object or a feature. A **point** may be represented by a single cell, a **line** by a sequence of neighbouring cells and a **polygon** by a collection of contiguous cells.

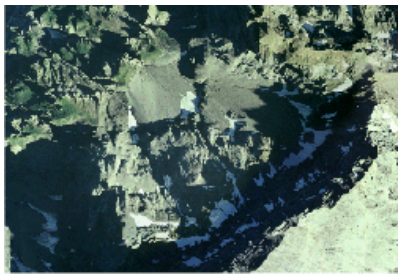
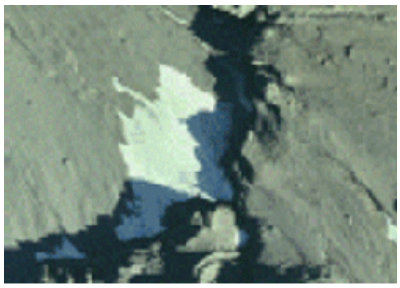
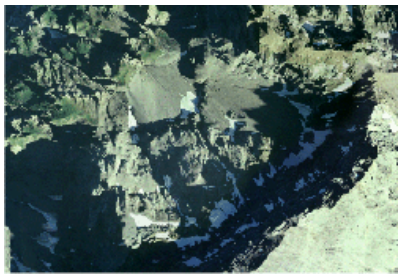
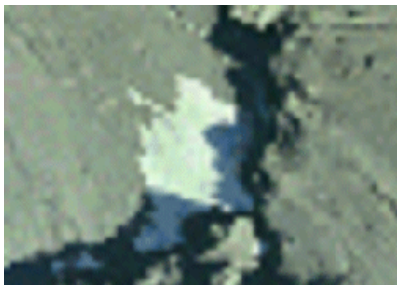
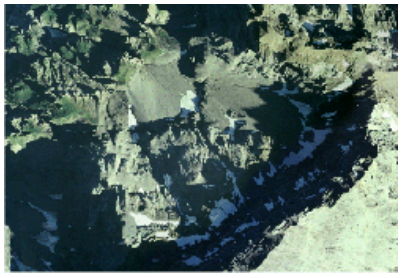
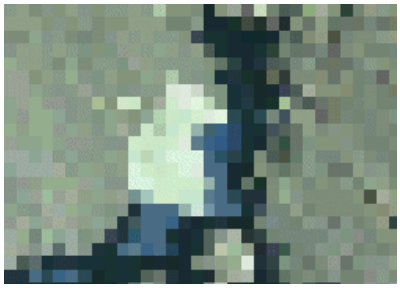


All cells in a raster must be the same size, determining the resolution. The cells can be any size, but they should be small enough to accomplish the most detailed analysis. A cell can represent a square kilometer, a square meter, or even a square centimeter.
Cells are arranged in rows and columns, an arrangement that produces a Cartesian matrix. The rows of the matrix are parallel to the x-axis of the Cartesian plane, and the columns to the y-axis. Each cell has a unique row and column address.



Resolution and storage size

Resolution can affect the data storage size. Storage requirements increase by the square of the image dimensions.

 *800 x 545 pixels 325KB*		 *Zoom in by 5*
 *400 x 272 pixels 91KB*		 *Zoom in by 5*
 *200 x 136 pixels 26KB*		 *Zoom in by 5*

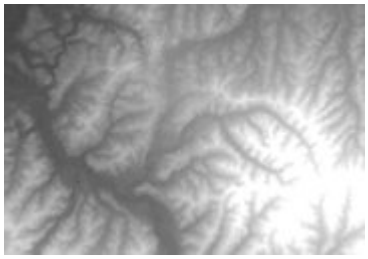
1.3.2. Raster Data Storage

Values may be stored as:

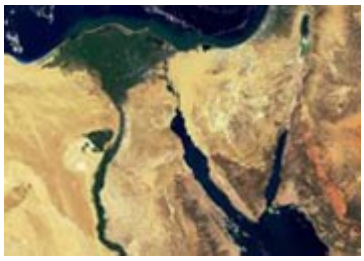
- Integer or
- Floating point.

In the case of floating points, pixels are representing continuous data. Remember that floating point numbers typically use up more storage space.
A raster can represent thematic data or raster images. Basically data structure is identical in both categories.

Thematic data

e.g. Land use	Digital elevation model
	

Raster images

Satellite images	Aerial fotographs
	
Orthophotos	Maps
	

1.3.3. Raster Data Compression

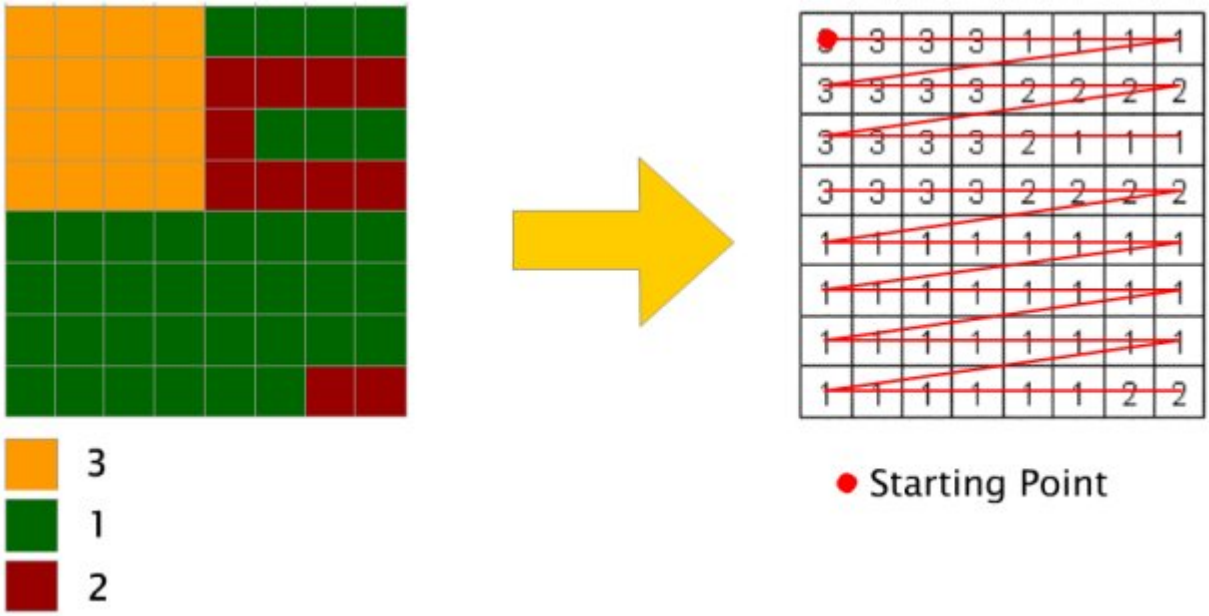
We can distinguish different ways of storing raster data, which basically vary in storage size and consequently in the geometric organisation of the storage. The following types of geometric elements are identified:

- Lines
- Stripes

- Tiles
- Areas (e.g. Quad-trees)
- Hierarchy of resolution

Raster data are managed easily in computers; all commonly used programming languages well support array handling. However, a raster when stored in a raw state with no compression can be extremely inefficient in terms of computer storage space.
As already said the way of improving raster space efficiency is data compression.
Illustrations and short texts are used to describe different methods of raster data storage and raster data compression techniques.

Full raster coding (no-compression)
By convention, raster data is normally stored row by row from the top left corner.



Example: The Swiss Digital elevation model (DHM25-Matrixmodell in decimeters)
In the header file are stored the following information:

- The dimension of the matrix (number of columns and rows)
- The x-coordinates of the South-West (lower left) corner
- The y-coordinates of the South-West (lower left) corner
- The cell size
- the code used for no data (i.e. missing) values

```
ncols 480 nrows 450 xllcorner 878923 yllcorner 207345 cellsize 25
 nodata_value -9999 6855 6855 6855 6851 6851 6837 6824 6815 6808
 6855 6857 6858 6858 6850 6839 6826 6814 6809 6854 6863 6865 6865
 6849 6840 6826 6812 6803 6853 6852 6873 6886 6886 6853 6822 6804
 6748 6847 6848 6886 6902 6904 6855 6808 6762 6686 6850 6859 6903
 6903 6881 6806 6739 6681 6615 6845 6857 6879 6856 6795 6706 6638
 6589 6539 6801 6827 6825 6769 6670 6597 6562 6522 6497 6736 6760
 6735 6661 6592 6546 6517 6492 6487 ...
```

PROBLEM: Big amount of data!

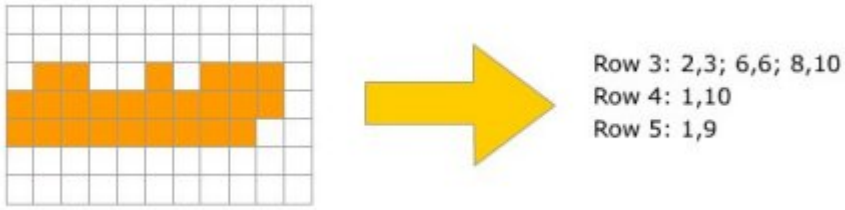
Runlength coding (lossless)

Geographical data tends to be "spatially autocorrelated", meaning that objects which are close to each other tend to have similar attributes:

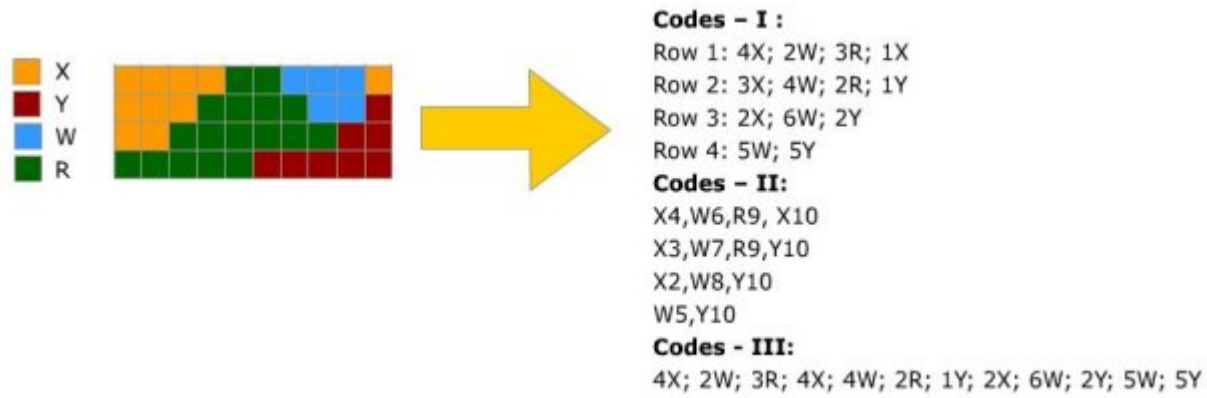
"*All things are related, but nearby things are more related than distant things*" (Tobler 1970) Because of this principle, we expect neighboring pixels to have similar values. Therefore, instead of repeating pixel values, we can code the raster as pairs of numbers - (run length, value).

The runlength coding is a widely used compression technique for raster data. The primary data elements are pairs of values or tuples, consisting of a pixel value and a repetition count which specifies the number of pixels in the run. Data are built by reading successively row by row through the raster, creating a new tuple every time the pixel value changes or the end of the row is reached.

Describes the interior of an area by run-lengths, instead of the boundary.



In multiple attribute case there a more options available:



We can note in *Codes - III*, that if a run is not required to break at the end of each line we can compress data even further.

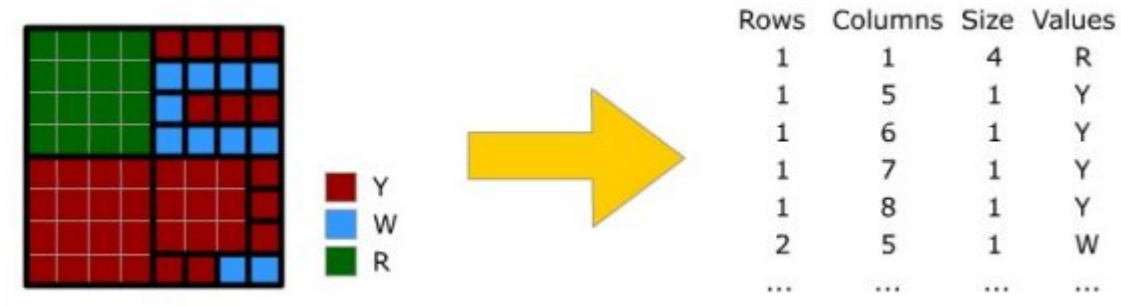
NOTE: run coding would be of little use for DEM data or any other type of data where neighboring pixels almost always have different values.

Chain coding (lossless)

See Rasterising vector data and the Freeman coding

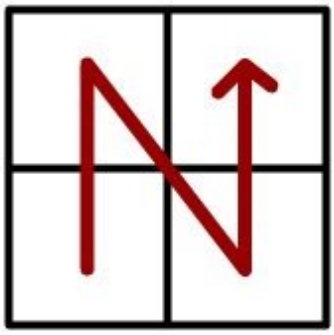
Blockwise coding (lossless)

This method is a generalization of run-length encoding to two dimensions. Instead of sequences of 0s or 1s, square blocks are counted. For each square the position, the size and, the contents of the pixels are stored.



Quandtree coding (lossless)

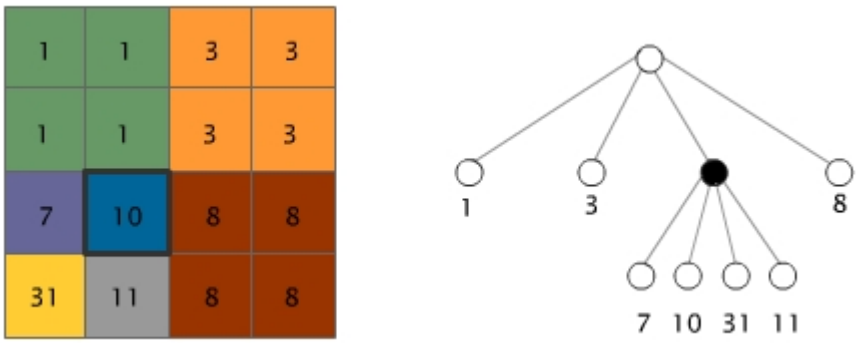
The quadtree compression technique is the most common compression method applied to raster data. Quadtree coding stores the information by subdividing a square region into quadrants, each of which may be further subdivided in squares until the contents of the cells have the same values.



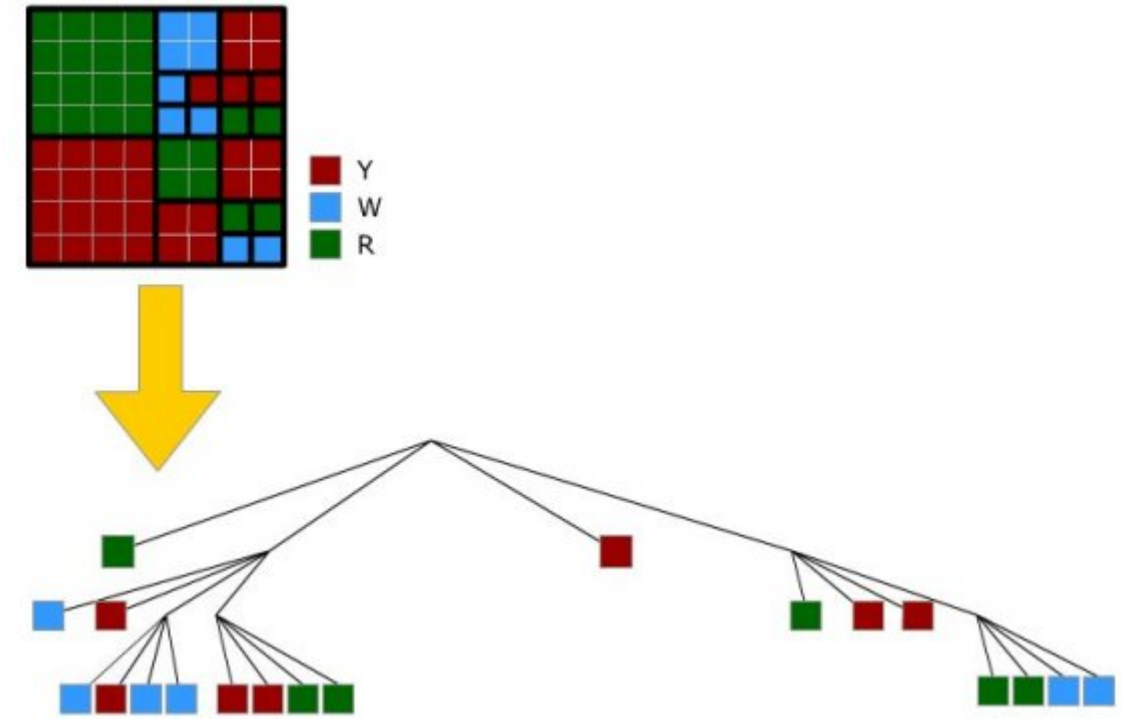
Reading versus

Example 1:

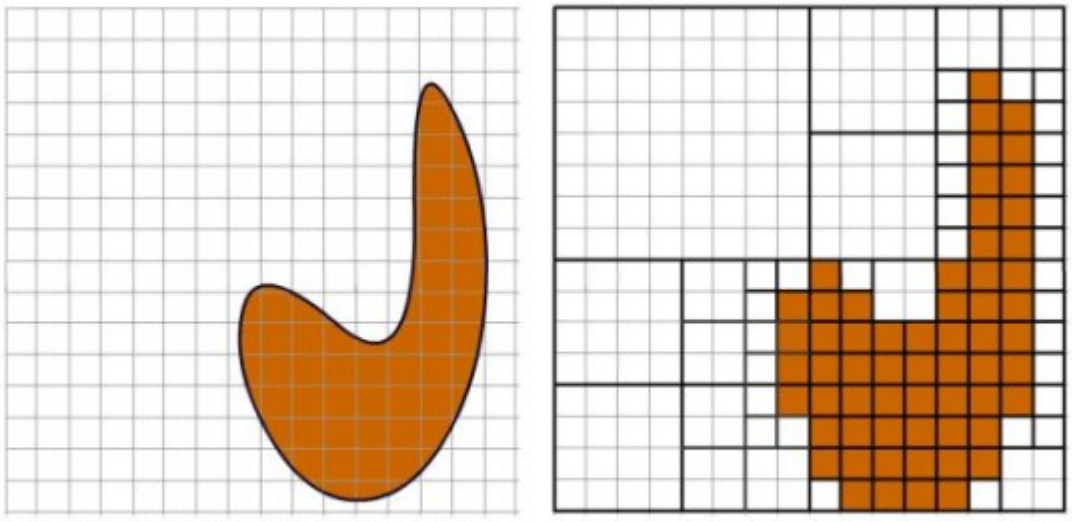
Positioncode of 10: 3,2



Example 2:

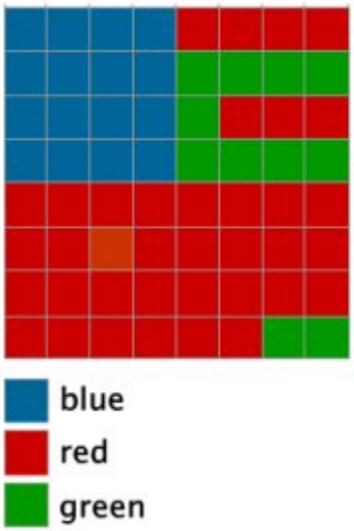


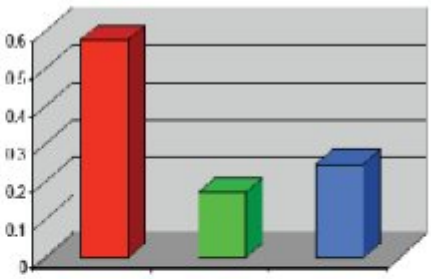
On the following figure we can see how an area is represented on a map and the corresponding quadtree representation. More information on constructing and addressing quadtrees, see Lesson "Spatial partitioning and indexing", Unit 2



Huffman coding (lossless compression)

Huffmann coding compression technique involves preliminary analysis of the frequency of occurrence of symbols. Huffman technique creates, for each symbol, a binary data code, the length of which is inversely related to the frequency of occurrence.





Color	$p(x_i)$	Code
red	0.58	0
blue	0.25	10
green	0.17	11

LZ77 method (lossless compression)

LZ77 compression is a loss-less compression method, meaning that the values in your raster are not changed. Abraham Lempel and Jacob Ziv first introduced this compression method in 1977. The theory behind this compression method is relatively simple: when you find a match (a data value that has already been seen in the input file) instead of writing the actual value, the position and length (number of bytes) of the value is written to the output (the length and offset - where it is and how long it is).

Some image-compression methods often referred to as LZ (Lempel Ziv) and its variants such as LZW (Lempel Ziv Welch). With this method, a previous analysis of the data is not required. This makes LZ77 Method applicable to all the raster data types.

JPEG-Compression (lossy compression)

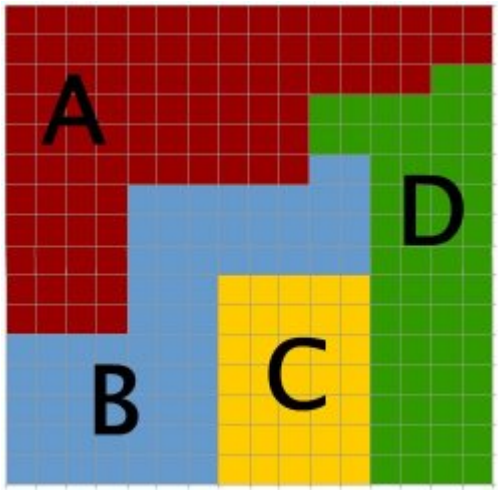
The JPEG-compression process:

- The representation of the colors in the image is converted from RGB to YCbCr, consisting of one luma component (Y), representing brightness, and two chroma components, Cb and Cr), representing color. This step is sometimes skipped.
- The resolution of the chroma data is reduced, usually by a factor of 2. This reflects the fact that the eye is less sensitive to fine color details than to fine brightness details.
- The image is split into blocks of 8×8 pixels, and for each block, each of the Y, Cb, and Cr data undergoes a discrete cosine transform (DCT). A DCT is similar to a Fourier transform in the sense that it produces a kind of spatial frequency spectrum.
- The amplitudes of the frequency components are quantized. Human vision is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Therefore, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components. The quality setting of the encoder (for example 50 or 95 on a scale of 0–100 in the Independent JPEG Group's library) affects to what extent the resolution of each frequency component is reduced. If an excessively low quality setting is used, the high-frequency components are discarded altogether.
- The resulting data for all 8×8 blocks is further compressed with a loss-less algorithm, a variant of Huffman encoding.

JPEG-Compression (left to right: decreasing quality setting results in a 8x8 block generation of pixels)
Text and graphic from: Wikipedia 2010

Example of runlength coding

In the following example, we can see the difference in data storing, between an uncompressed raster file and a compressed one.



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

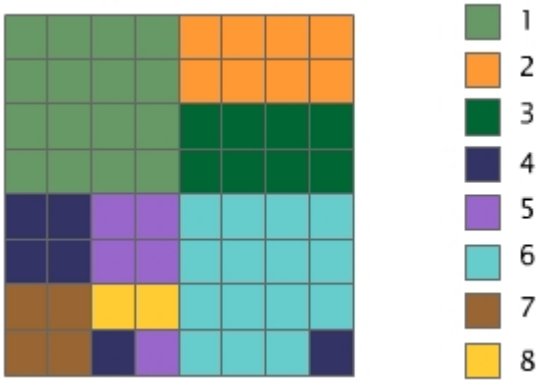
Full raster coding 256 values

VALUE	LENGHT	ROW
A	16	0
A	16	1
A	14	2
D	2	2
A	10	3
D	6	3
A	10	4
D	6	4
A	10	5
B	2	5
D	4	5
A	4	6
B	8	6
D	4	6
A	4	7
B	8	7
D	4	7
A	4	8
B	8	8
D	4	8
A	4	9
B	3	9
C	5	9
D	4	9
A	4	10
B	3	10
C	5	10
D	4	10
B	7	11
C	5	11
D	4	11
B	7	12
C	5	12
D	4	12
B	7	13
C	5	13
D	4	13
B	7	14
C	5	14
D	4	14
B	7	15
C	5	15
D	4	15

runlength coding 132 values

Proof your knowledge

Compress the following raster file with the compression techniques that you have learnt in this unit.
How much space is needed for storage, if we assume that bytes (8 bits) can be used to store the data values?



1.4. Image Formats

Overview

Typically, a digital image is a representation of a two-dimensional image as a finite set of digital values, called pixels. These are held in computer memory as a two-dimensional array of small integers, which are transmitted or stored often in a compressed form. Image file formats provide a standardized method of organizing and storing image data.
In this unit an overview on different image data formats is given, summarizing some characteristics and compression properties.

1.4.1. Digital Image Formats

File formats
File formats are intended to store particular kinds of digital information: the JPEG format, for example, is designed only to store still images, while the GIF format supports storage of both still images and simple animations. The most well-known formats have file specifications that describe exactly how the data is to be encoded.
Since files are seen by programs as streams of data (0s and 1s), a method is required to determine the format of a particular file within the file system. One popular method is to determine the format based on the final portion of the filename, known as the filename extension.

Digital images
The standard greyscale images use 256 shades of grey from 0 (black) to 255 (white).
With color images, the situation is more complex. For a given number of pixels, considerably more data is required to represent the image and more than one color model is used.
We will consider the following data format:

- GIF
- JPEG
- TIFF
- BMP
- PNG

Image formats

GIF (Graphic Interchange Format)
GIF uses lossless LZW compression for relatively small file sizes, as compared to uncompressed data. GIF files offer optimum compression (smallest files) for solid color graphics, because objects of one exact color compress very efficiently in LZW. The LZW compression is lossless, but of course, the conversion to only 256 colors may be a great loss.
The GIF format can only show 256 colors but it can choose any of the 16 million colors in a 24-bit image.
8 bit = 256 colors
7 bit = 128 colors
6 bit = 64 colors
5 bit = 32 colors

4 bit = 16 colors
3 bit = 8 colors
2 bit = 4 colors
1 bit = 2 colors (usually B/W)
GIF optionally offers transparent backgrounds, where one palette color is declared transparent, so that the background can show through it.

TIFF (Tagged Image File Format)
TIFF files have many formats: Black and white, greyscale, 4- and 8-bit color, full color (24-bit) images. TIFF files support the use of data compression using LZW and other compression standards.

BMP (Bitmap)
B/W Bitmap is monochrome and the color table contains two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the color table. If the bit is set, the pixel has the color of the second entry in the table.
4-bit Bitmap has a maximum of 16 colors. Each pixel in the bitmap is represented by a 4-bit index into the color table array. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8-bit Bitmap has a maximum of 256 colors. Each pixel in the bitmap is represented by a 1-byte index into the color table.
24-bit Bitmap has a maximum of 2^24 colors. Each pixel is a 3-byte sequence in the bitmap array represents the relative intensities of red, green, and blue.

JPEG (Joint Photographic Expert Group Format)
A JPEG image provides very good compression, but does not uncompress exactly as it was; JPEG is a lossy compression techniques.
Compressions up to 50:1 are easily obtainable.

PNG (Portable Network Graphics)
The PNG format was designed to replace the antiquated GIF format, and to some extent, the TIFF format. It utilizes lossless compression. It is a universal format that is recognized by the World Wide Web consortium, and supported by modern web browsers.

Summarizing tables

Supported data types

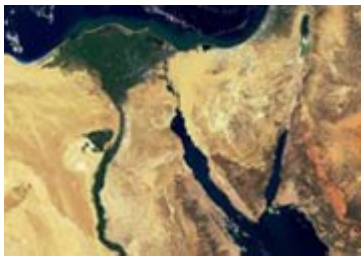
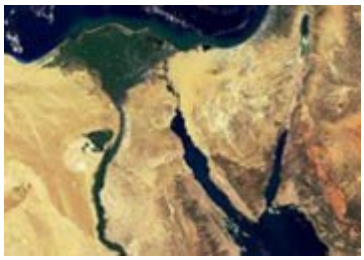
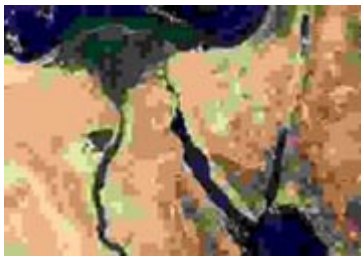
Format	Supported data types
BMP	1/4/8-bit unsigned integer
GIF	8-bit unsigned integer
JPEG	8/16-bit unsigned integer
TIFF	unsigned integer
	4-bit unsigned integer
	8-bit unsigned integer
	16-bit unsigned integer
	32-bit unsigned integer
	8-bit signed integer
	16-bit signed integer
	32-bit signed integer
	32-bit floating point
PNG	1-bit unsigned integer
	2-bit unsigned integer
	4-bit unsigned integer
	8-bit unsigned integer
	16-bit unsigned integer

Here are some representative file sizes for 1943 x 1702 pixels image (24-bit RGB), which can give you an idea of file sizes:

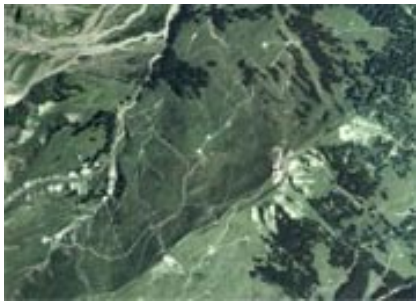
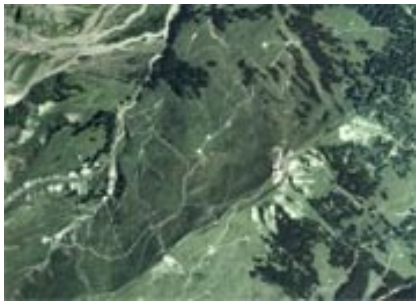
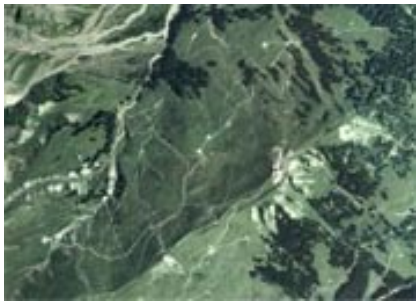
File type	File size	Description
TIFF	9.9 MB	
TIFF LZW	8.4 MB	
PNG	6.5 MB	PNG files are about 25% small then TIFF and about 10% to 30% smaller than GIF files for indexed data.
JPEG	1.0 MB	The JPEG format compresses the picture every time it is saved and that means the file size is reduced every time it is saved.
BMP	9.9 MB	

The table below represents different compression rates of the same techniques. Also in this case you can see the differences in file sizes.

JPEG Compression (483 x 337 pixels)

 *No compression 71 KB*	 *50% 38 KB*	 *99% 5 KB*
 *B/W no compression 16 KB*		

TIFF Compression (1108 x 798 pixels)

		
No compression 2592 KB	*LWZ compression 2154 KB*	*Huffman conding 206 KB*

1.5. Summary

Efficient data storage and compression is a main factor working with spatial data. This lesson showed the principles how vector and particularly raster data is stored on computers and the different possibilities to compress data, either lossless or lossy, were discussed. Additionally, some of the most widely used image formats were considered and their properties as well as their advantages and disadvantages were outlined.

1.6. Bibliography

- **Longley, P. et al**, 2001. *Geographic Information Systems and Science*. Chichester: Wiley.
- **Tobler, W.R.** , 1970. A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46, 234-240.