1. Graphical primitives and canvas.

Pre-requirements

1. Java programming language.

Student will learn how to create JWindow, add JPanel into it and use paintCompontent method to plot graphical primitives.

Code example:

```
class MyPanel extends JPanel
{
  public void paintComponent(Graphics g)
  {
    Graphics2D g2 = (Graphics2D)g;
    super.paintComponent(g2);

    Point2D center = new Point2D.Float(0.0f, 0.0f);
    Line2D line = new Line2D.Float(center, center);

    GeneralPath p = new GeneralPath();
    p.moveTo(-100.0f,  -25.0f);
    p.lineTo( 100.0f,  -25.0f);
    p.lineTo( -50.0f,  100.0f);
    p.lineTo(   0.0f, -100.0f);
    p.lineTo(  50.0f,  100.0f);
    p.closePath();

    g2.translate(200.0f, 200.0f);
    g2.rotate(3.14159f/6.0f);
    g2.scale(2.0f, 2.0f);

    g2.setColor(Color.black);
    g2.setStroke(new BasicStroke(6.0f));
    g2.draw(line);
    g2.draw(p);
  }
}
```

Task exercise

Implement a simple program drawing on the screen a line, rectangle and circle. The user should be able to provide the parameters of primitives using the text box and click the mouse points by making modifications to the appearance of the primitive .

2. PPM file format

Student will learn how to draw pixels on BufferedImage object and load image's pixel definitions from a PPM file.

PPM file format exists in two flavours P3 and P6. We assume that the images are in color and not a gray scale: if the maximum color value is less than 256 to one pixel consists of 3 bytes, R component, G component and the B component. If the maximum color value is greater than one byte can accommodate (255), but less than 65536 (ie. two bytes), then one pixel is composed of 6 bytes, thus two bytes per component, and each component is most significant byte first. Specifically, we assume that there will be more colors than 65536. Example:

```
P6
# some comment
3 2
255
!@#$%^&*()_+|{}:"<
```

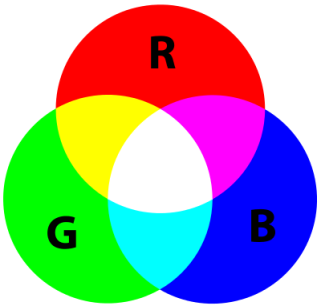
The file above represents an image 3 columns * 2 lines * 3 components = 18 bytes of storage, more specifically:

```
value:     !@#$%^&*()_+|{}:"<
component: RGBRGBRGBRGBRGBRGB
row:       111111111222222222
column:    111222333111222333
```

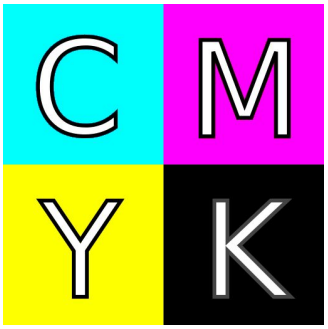
Please note that the colors should be scaled linearly, when the maximum value eg. 15, in our case before displaying must be scaled to 0-255, similarly the maximum value eg. 13200 scaling must also be made. When writing to the JPG you lose quality through compression. Format PPM has no compression, the data represent exactly how the image looks.

Task exercise:
Implement functionality of reading and displaying image files in PPM format P6 and P3. Note the error handling (eg. your program must support also damaged files) and the performance of the reading algorithm (using buffers and big blocks rather than byte by byte). The user should also be able to load and save JPEG files.

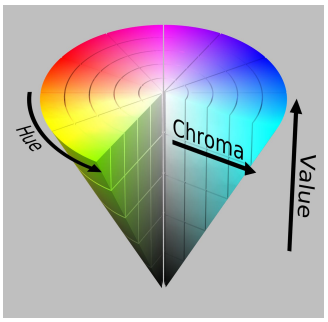
3. Color models (RGB, CMYK, HSV)



The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography.



The CMYK color model (process color, four color) is a subtractive color model, used in color printing, and is also used to describe the printing process itself. CMYK refers to the four inks used in some color printing: cyan, magenta, yellow, and key (black). Though it varies by print house, press operator, press manufacturer, and press run, ink is typically applied in the order of the abbreviation.
The "K" in CMYK stands for key because in four-color printing, cyan, magenta, and yellow printing plates are carefully keyed, or aligned, with the key of the black key plate.



HSL and HSV are the two most common cylindrical-coordinate representations of points in an RGB color model. The two representations rearrange the geometry of RGB in an attempt to be more intuitive and perceptually relevant than the cartesian (cube) representation. Developed in the 1970s for computer graphics applications, HSL and HSV are used today in color pickers, in image editing software, and less commonly in image analysis and computer vision.

Task exercise:
Color spaces. Implement program:
a. allowing color space conversion from RGB to CMYK and CMYK to RGB. Additionally implement the presentation of the user-selected color.

RGB --> CMYK	CMYK --> RGB
Black = minimum(1-Red,1-Green,1-Blue) Cyan = (1-Red-Black)/(1-Black) Magenta = (1-Green-Black)/(1-Black) Yellow = (1-Blue-Black)/(1-Black)	Red = 1-minimum(1,Cyan*(1-Black)+Black) Green = 1-minimum(1,Magenta*(1-Black)+Black) Blue = 1-minimum(1,Yellow*(1-Black)+Black)

b. draw RGB cube and HSV cone

4. Pixel-in-point transformations. Methods that improve image quality.

Pixel-in-point transformations are algorithms that are working only on each pixel, not pixels' surroundings.

Grayscale

The intensity of a pixel is expressed within a given range between a minimum and a maximum, inclusive. This range is represented in an abstract way as a range from 0 (total absence, black) and 1 (total presence, white), with any fractional values in between.

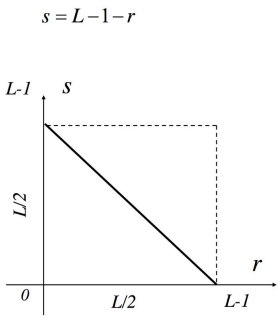
To convert a color from a colorspace based on an RGB color model to a grayscale representation of its luminance, weighted sums must be calculated in a linear RGB space, that is, after the gamma compression function has been removed first via gamma expansion:

$$Y = 0.2126R + 0.7152G + 0.0722B$$

Luma coding in video streams:

$$Y' = 0.299R' + 0.587G' + 0.114B'$$

Negative

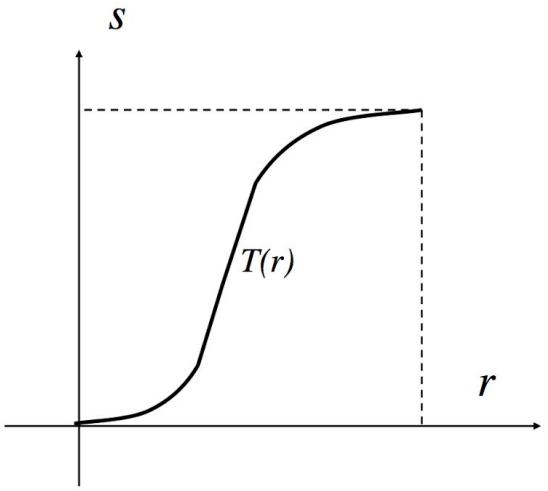


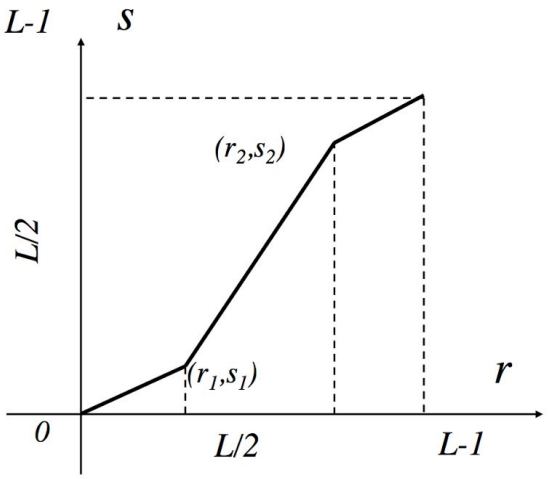
Task exercise:
Please implement a program that performs basic transformation point in the images: negative, addition, subtraction, multiplication, division, change of the brightness, the transition to the gray scale (both equations) .

5. Histogram presentation, histogram normalisation and equalisation.

Definitions:
f(x,y) – input image
g(x,y) – output image
g(x,y)=T[f(x,y)]
r=f(x,y),s=g(x,y),s=T(r)

Contrast stretching





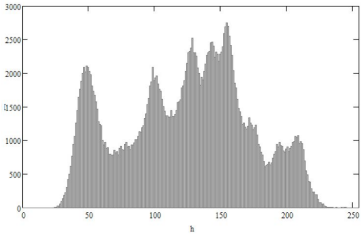
$$s = \begin{cases} a \cdot r & 0 \le r < r_1 \\ b \cdot (r - r_1) + s_1 & r_1 \le r < r_2 \\ c \cdot (r - r_2) + s_2 & r_2 \le r < L \end{cases}$$

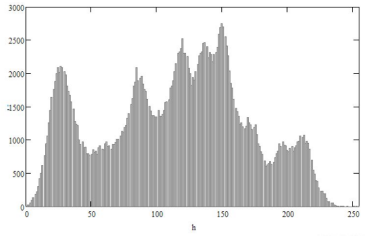
$$a = \frac{s_1}{r_1} \qquad b = \frac{s_2 - s_1}{r_2 - r_1}$$

$$c = \frac{L - 1 - s_2}{L - 1 - r_2}$$

Histogram stretching

$$S_k' = \frac{S_k - k_{\min}}{k_{\max} - k_{\min}} Z_k, \qquad k = 0,1,2,...,L-1$$





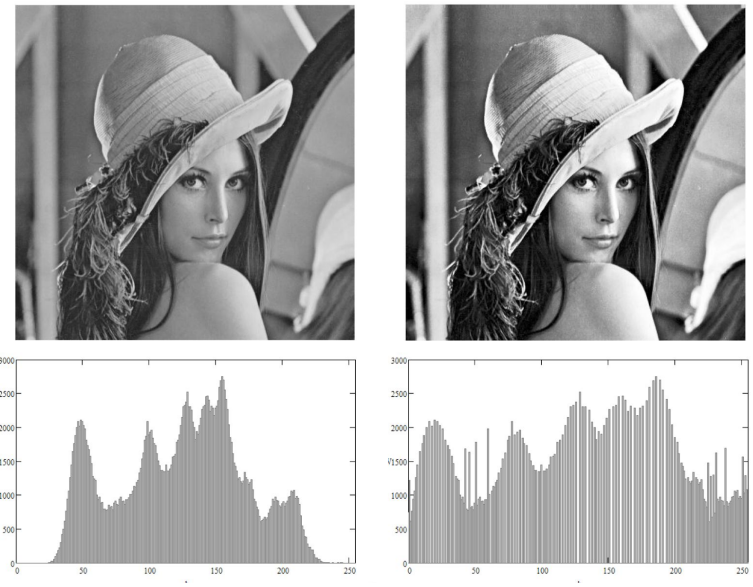
Histogram equalize

Let the image has L different shades of gray rk , for k = 0,1, ... , L - 1 and let nk represents the number of pixels of shade rk.

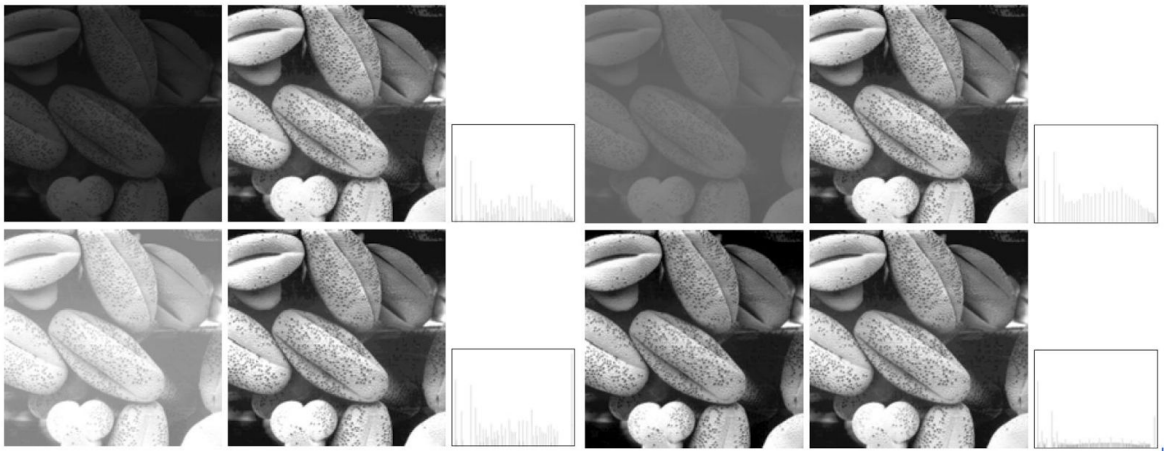
$$S_k = \sum_{i=0}^{k} p_r(r_k) = \sum_{i=0}^{k} \frac{n_k}{n}, \qquad k = 0,1,2,...,L-1$$

where pr (rk) is the probability of rk and described with an equation:

$$p_r(r_k) = \frac{n_k}{n}, \qquad k = 0,1,2,...,L-1$$



Examples:



Task exercise:
Please implement program that can import image and draw its histogram. It should be also possible to make linear normalisation of the histogram by contrast stretching and histogram equalisation.

7. Binarization and methods for automatic threshold level selection.

Binary Threshold Level Selection[1]

When creating a binary image having only two intensity levels (black and white) from an original grayscale digital image that has 256 possible intensity values (for an 8-bit image), a binary threshold level must be chosen to designate the intensity level at which binary segregation occurs. This interactive tutorial explores the use of various algorithms utilized in the methodology for choosing a single binary threshold level.

When a large number of similar specimen images are captured under identical conditions, then it is often feasible to choose a single threshold level based on a subset of the images and then to apply this threshold level to all of the images in the collection. A similar approach involves segmenting a group of images according to a fixed ratio of black pixels to white pixels, with the ratio being selected appropriate to the particular image set.

A simple algorithm for determining the threshold level for a given image (and the percentage of black pixels desired) operates by computing the smallest nonnegative integer **K** such that the following relation is satisfied:

$$\sum_{i=0}^{K} h[i] \geq N \cdot p$$

In the equation above, **N** represents the total number of pixels in the image, **p** represents the percentage of black pixels desired, and **h** represents the image histogram sequence. This method is automatic in the sense that the threshold gray-level for a particular image will be calculated algorithmically, but it is not automatic in the sense that the user must choose the percentage of black pixels desired.

One truly automatic algorithm for choosing a binary threshold level is known as iterative selection. This technique employs an algorithm that sequentially refines an initial estimate of a suitable threshold level. After the threshold level has been estimated, iterative selection operates by segregating the image pixels into specimen and background classes. The algorithm then utilizes the gray levels contained in each class to improve the initial threshold level estimate, which is the mean gray level of the image. On each iteration, the mean gray level for all pixels below the threshold is determined, and is denoted as T(B). The mean gray level for all pixels greater than or equal to the threshold level is also determined, and is denoted as T(W). The new estimate is computed as the arithmetic mean or average of T(B) and T(W), or (T(B) + T(W)) / 2. A general equation for computing the kth estimate of the threshold using the image histogram h is:

[1] http://olympus.magnet.fsu.edu/primer/java/digitalimaging/processing/automaticthresholding/index.html

$$T_k = \frac{\sum_{i=0}^{T_{k-1}} i \cdot h[i]}{2\sum_{i=0}^{T_{k-1}} h[i]} + \frac{\sum_{j=T_{k-1}+1}^{N} j \cdot h[j]}{2\sum_{j=T_{k-1}+1}^{N} h[j]}$$

In this equation, T(0) represents the initial estimate of the threshold level. The algorithm terminates when T(B) and T(W) are equal, or when T(k) = T(k - 1).

Another method of automatic threshold selection is based on viewing the gray-level histogram of an image as an estimated probability density function of the gray-levels comprising specimen and background pixels. If the specimen pixels and background pixels in the image are each considered to be normally distributed but contained in separate classes, then the gray-level histogram can be seen as an approximation to the sum of two normal distributions given by the equation:

$$p(g) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-((g-\mu_1)^2/2\sigma^2_1)} + \frac{1}{\sigma_2\sqrt{2\pi}} e^{-((g-\mu_2)^2/2\sigma^2_2)}$$

In the equation above, μ(k) and σ(k) (for k = 1, 2) represent the mean and variance, respectively, of gaussian distributions that approximate the specimen and background pixel distributions. Minimum error binary segmentation is an algorithm that operates by arbitrarily dividing the histogram into two parts, modeling each part with a normal distribution, and comparing the combined models with the original histogram. The threshold level that minimizes the discrepancy between the model and the histogram is the one that displays a minimum error.

Other methods of automatic binary segmentation rely on the concept of **entropy**, a term describing a measure of information content. In this sense, entropy represents a quantitative description of the amount of information in a message based on the logarithm of the number of the possible equivalent messages.

Task exercise:

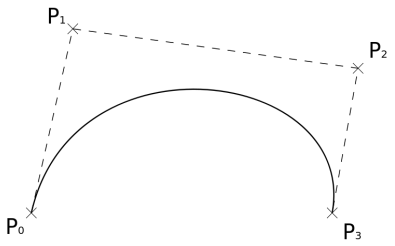
Please implement a function of performing binarization on images and automatic threshold level selection :

a. manually by the user
b. Percent Black Selection
c. Mean Iterative Selection
d. Entropy Selection
e. Minimum Error
f. Fuzzy Minimum Error

8. Bézier curve[2]

A Bézier curve is a parametric curve frequently used in computer graphics and related fields. Generalizations of Bézier curves to higher dimensions are called Bézier surfaces, of which the Bézier triangle is a special case.

In vector graphics, Bézier curves are used to model smooth curves that can be scaled indefinitely. "Paths", as they are commonly referred to in image manipulation programs are combinations of linked Bézier curves. Paths are not bound by the limits of rasterized images and are intuitive to modify.



The general formula Bezier curve:

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i, t \in [0,1]$$

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, i = 0, ...n$$

The formula for the curve of the third degree:

$$P_x(t) = A_x(1-t)^3 + 3B_x t(1-t)^2 + 3C_x t^2(1-t) + D_x t^3, 0 \leqslant t \leqslant 1$$
$$P_y(t) = A_y(1-t)^3 + 3B_y t(1-t)^2 + 3C_y t^2(1-t) + D_y t^3, 0 \leqslant t \leqslant 1$$

Task exercise

Please implement a program that draws Bezier curve. The user should be able to modify any checkpoints. In addition, the algorithm will be scored Bezier curve drawing any grade.

[2] wikipedia

9-10. 2D transformations[3]

Most common geometric transformations that keep the origin fixed are linear, including rotation, scaling, shearing, reflection, and orthogonal projection; if an affine transformation is not a pure translation it keeps some point fixed, and that point can be chosen as origin to make the transformation linear. In two dimensions, linear transformations can be represented using a 2×2 transformation matrix.

Rotation

For rotation by an angle θ **clockwise** about the origin the functional form is $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$. Written in matrix form, this becomes:[4]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Similarly, for a rotation **counter-clockwise** about the origin, the functional form is $x' = x\cos\theta - y\sin\theta$ and $y' = x\sin\theta + y\cos\theta$ and the matrix form is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

These formulae assume that the *x* axis points right and the *y* axis points up. In formats such as SVG where the *y* axis points down, these matrices must be swapped.

Shearing

For shear mapping (visually similar to slanting), there are two possibilities.

A shear parallel to the *x* axis has $x' = x + ky$ and $y' = y$. Written in matrix form, this becomes:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

A shear parallel to the *y* axis has $x' = x$ and $y' = y + kx$, which has matrix form:

---

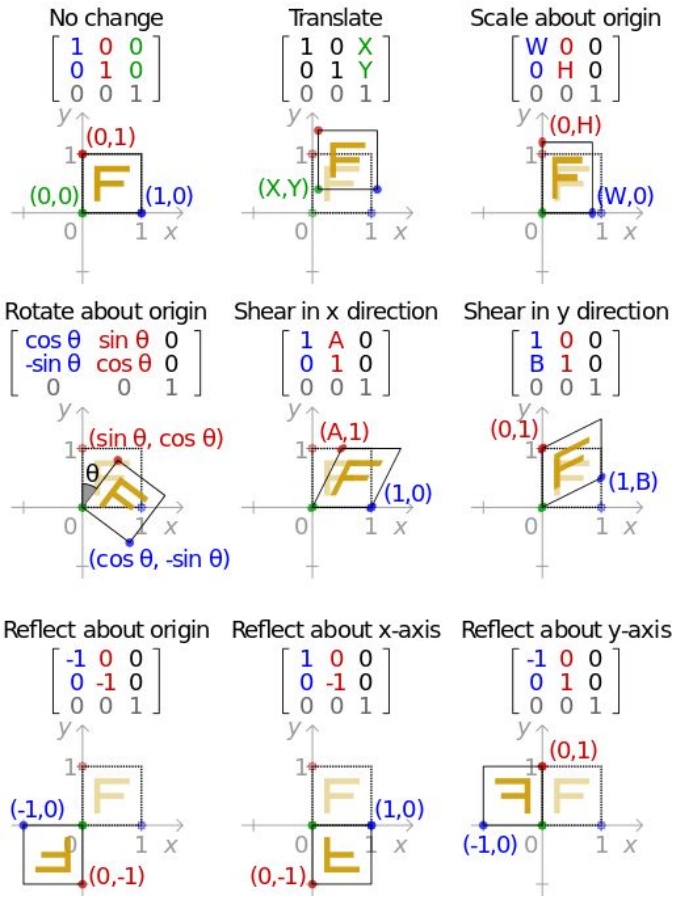
[3] wikipedia

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reflection

To reflect a vector about a line that goes through the origin, let $\vec{l}=(l_x,l_y)$ be a vector in the direction of the line:

$$\mathbf{A} = \frac{1}{\|\vec{l}\|^2} \begin{bmatrix} l_x^2 - l_y^2 & 2l_x l_y \\ 2l_x l_y & l_y^2 - l_x^2 \end{bmatrix}$$

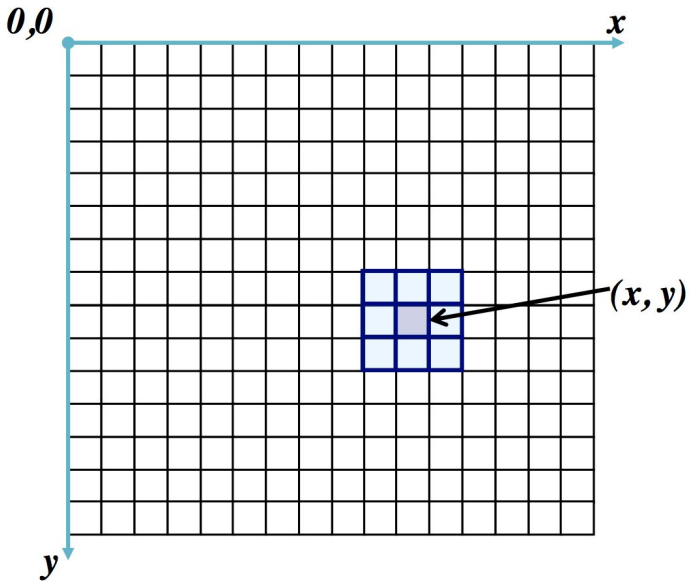


Task exercise:

2D Transformations. Implement a program to define a 2 - dimensional vector object and performing the transformation of a 2- dimensional vector on these objects using transformation matrices.

11. Image filters: mean, median, gaussian blur, sharpening

Neighbourhood operations perform calculations of the image in a pixel and neighboring pixels. Neighboring pixels are usually a rectangular surrounding the processed pixel. Surrounding neighborhood is called filter mask. It is possible that filter mask can have any shape and size.



Mean filter

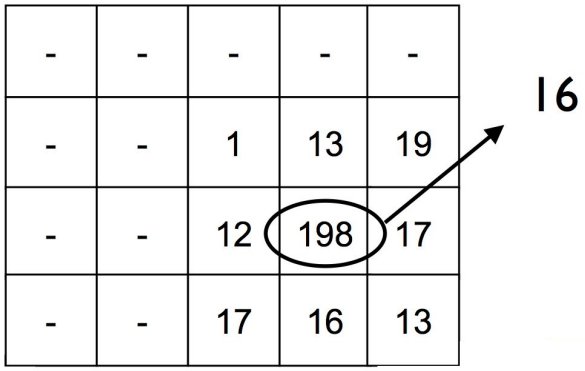
The two-dimensional convolution operation mask and matrix defining image, result are averaged values of all the pixels around the neighborhood value.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$$g(x,y) = \frac{\sum_{i=-m}^{m}\sum_{j=-m}^{m} w(i,j)\cdot f(x+i,y+j)}{\sum_{i=-m}^{m}\sum_{j=-m}^{m} w(i,j)}$$

Median filter

Center value in sorted vector of values.



Sobel edge detector

Sobel horizontal mask detects vertical edges, a vertical mask - horizontal edges

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Sharpening filters

Line Filters based on the second partial derivatives

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x+1, y) - f(x, y) - [f(x, y) - f(x-1, y)]}{1}$$

$$= f(x+1, y) + f(x-1, y) - 2f(x, y)$$

, the same for y.

Then:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Example filter masks for sharpening filters:

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Example image processing results

Edge detectors

-1	-1	-1
-1	8	-1
-1	-1	-1

Sharpening filters (highpass filter)

-1	-1	-1
-1	9	-1
-1	-1	-1

Lowpass filters

$$\sum w_{ij} = 9$$

0	1	0
1	1	0
0	1	0

Task exercise

Please implement a program that performs operations of filtration: smoothing filter (averaging), median filter, edge detection filter (sobel), high pass filter sharpen, Gaussian blur filter, convolution masks of any size

12. Morphological operators: hit-or-miss, dilation, erosion, opening, closing.

Mathematical morphology describes various techniques of image processing, which relate to the shape of the features in the image. This is the mathematical tool for the analysis of geometric structures in the image. Mathematical morphology is applicable to both binary images, as well as for grayscale images.

Geometrical structure is "discovered" by affecting the image of the so-called “structuring element”. These elements modify the shape of the object by specifying its final structure.

Structuring element

Structuring element in the case of image are defined structure of neighbouring pixels. Denote one point, called the initial (or central). The starting point indicates the pixel to which the result of the operation performed.

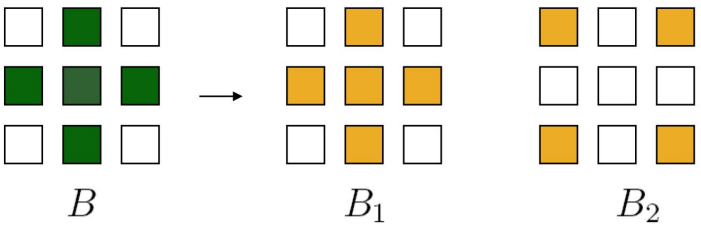
1	1	1
1	**1**	1
1	1	1

0	1	0
1	**1**	1
0	1	0

0	0	1	0	0
0	1	1	1	0
1	1	**1**	1	1
0	1	1	1	0
0	0	1	0	0

Hit-or-miss

Operation hit-or-miss is an essential tool for the detection of shape. Operation is used to search items in the image - shaped exactly like structuring element. Structuring element consists of two types of pixels B = (B1, B2), the object pixels and background pixels.



Hit-or-miss operation is a result of

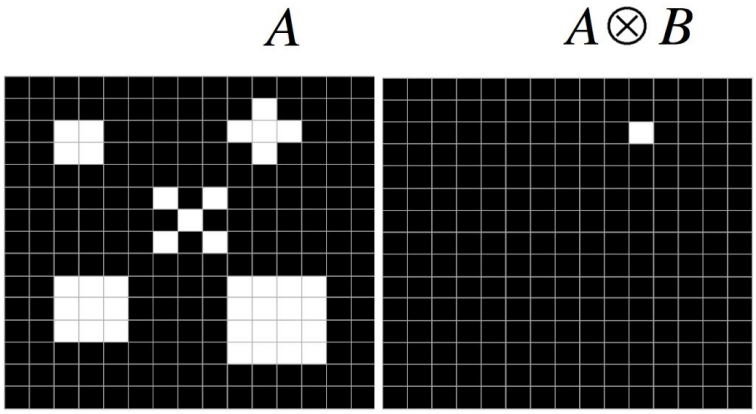
$$A \otimes B = \{A \ominus B_1 \cap A^c \ominus B_2\}$$

B_1

0	1	0
1	1	1
0	1	0

B_2

1	0	1
0	0	0
1	0	1

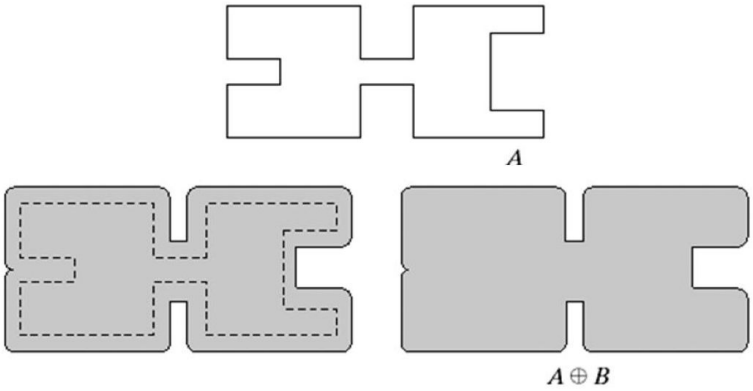


Dilation

For sets A and B in the area Z2 where B is the structuring element, a dilatation is defined as:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \phi\}$$

Dilation image A by the element B is the set of all displacements, such that the reflection B covers at least one element of the set A after translation.

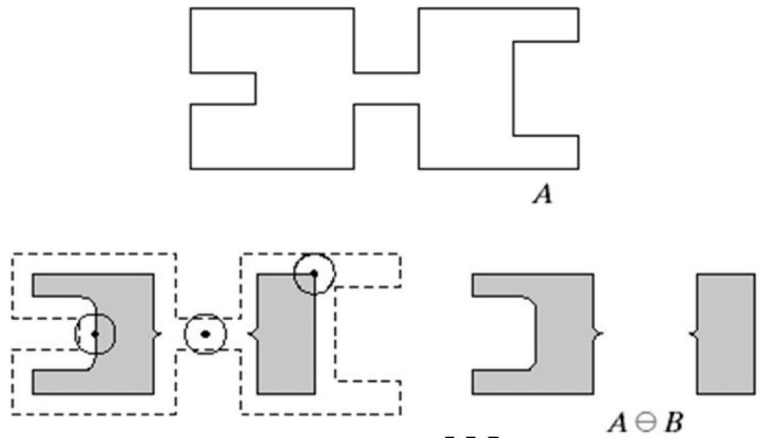


Images are from „Digital Image Processing ", Gonzalez and Woods

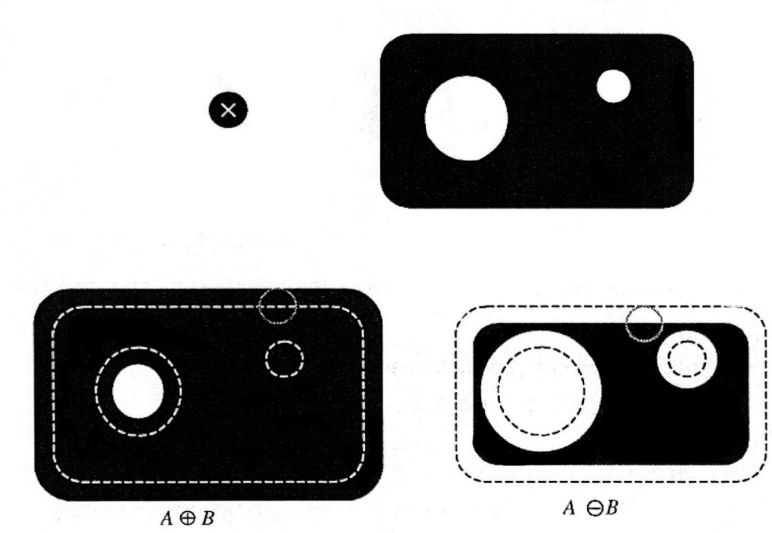
Erosion

For sets A and B in the area Z2 where B is the structuring element, an erosion is defined as:

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

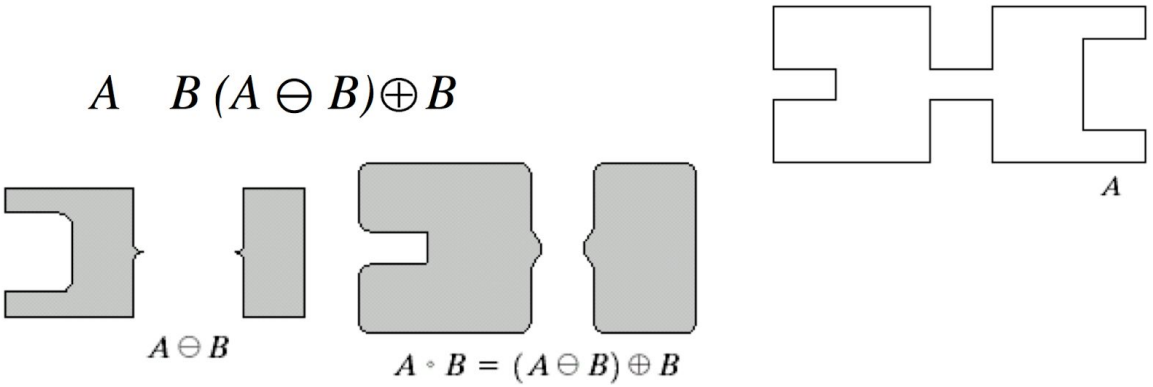


Dilation and Erosion example:



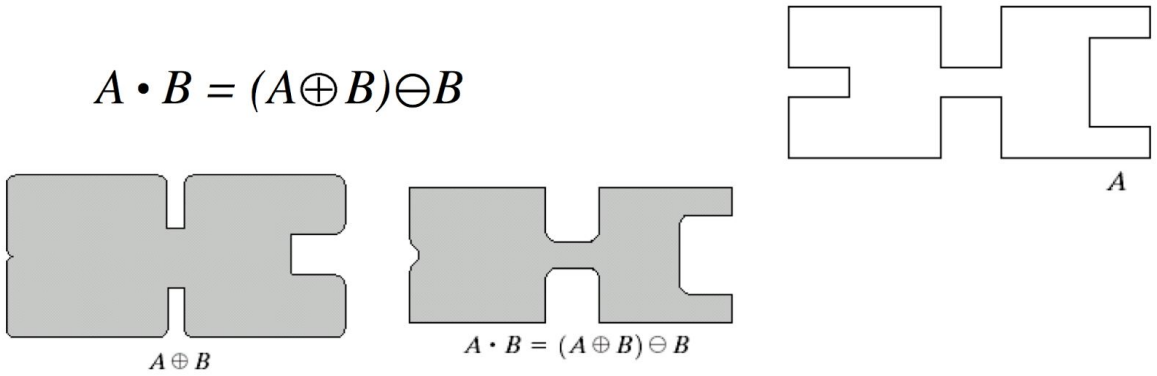
Opening

The operation of opening the image A by structuring element B is done by Erosion and the subsequent Dilation operation.



Closing

The operation of closing the image A by structuring element B is done by Dilation and the subsequent Erosion operation.



Task exercise

Morphological operators. Please implement a program that converts the image using morphological filters: hit-or-miss, dilation, erosion, opening, closing.

13. Image analysis example.

Implement application that calculates how much % of the space is covered by greenery on the campus of the Bialystok University of Technology on the basis of satellite images of the campus.

14-15. Final project implementation and exam.

The final project. Graphical editor with user interface, that uses implementations of the tasks 1-12. Please collect all of implemented functionalities into a single program with a user interface.