Image Analysis – Lecture No. 1

Syllabus for the Erasmus students

- Image segmentation (detecting objects, areas, edges).
- Computing attributes for object recognition.
- Tools for object recognition – neural networks, classifiers.
- Reconstructing scene from two or more images.
- Motion analysis. Analysis of the videosequences.

Contact:

Eduard Sojka
Room EB410
eduard.sojka@vsb.cz

Detecting areas and edges in images

Thresholding

Thresholding is the simplest way how to detect the areas (objects) in images. It is carried out according to the following formula

$$g(x,y) = \begin{cases} 1, & f(x,y) \geq t \\ 0, & \text{otherwise} \end{cases},$$

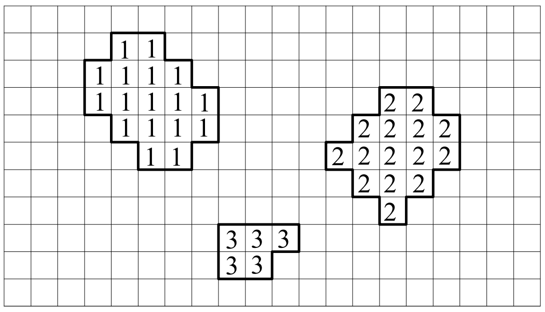
where *g*() is the output binary image in which the value of 1 indicates that the pixel belongs an object, and 0 indicates that the pixel belongs to the background. The thresholding can be carried out relatively easily. The only problem is determining the threshold. It need not be easy or even possible to determine the value that is suitable.

One possible way of how to determine the threshold is based on the analysis of the histogram. The suitable value often is in the valley between two hills in the histogram (naturally, it is not necessary that the histogram has such a shape – determining the threshold value is then usually more difficult).

Another possibility of how to determine the values of threshold is based on the use of probability theory.

Indexing objects

In practice, the thresholding is often followed by indexing the objects in images. During indexing, the objects are given unique numbers that are used in further processing the image, e.g., in recognising the objects. The result of indexing is shown in the following image.

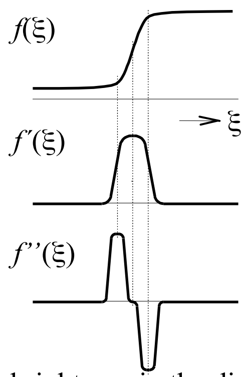


The result of indexing

I would like to ask you to devise your own algorithm for indexing as a home work. It should not be so difficult.

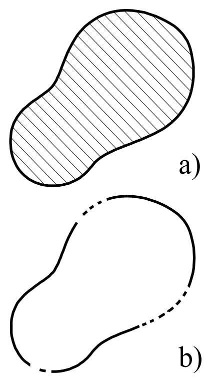
Detecting edges

The idea is that every object (for example the object that is to be recognised) is an area that is surrounded by its boundary. Boundary consists of particular points called edge points. The task is to determine these points and, in this way, also the whole boundary of the object. Let us consider the grey scale images. The edge points may be detected by analysing the brightness function. The following figure (upper image) shows a typical shape of the brightness function in the direction across the edge.



The brightness in the direction across the edge; its first and second derivatives, respectively

The theoretical edge point is usually expected to be at the inflexion point of the brightness function. Since every area should be enclosed with its boundary, it seems that detecting the whole areas and detecting edges should lead to the equivalent results. In practice, however, problems may occur. If the objects are mot well seen in the image or if the image is damaged by noise, it may happen that the edges and boundaries are not detected as closed loops, as we theoretically would like. On the other hand, the edges may also be detected at places where they are not present in reality. Due to these problems, the detection of the edges and whole areas need not be equivalent in practice.



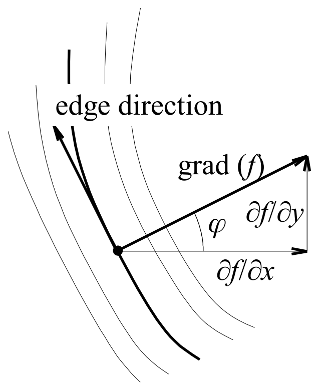
On the difference between the area and edge detection.

Gradient methods of edge detection

The methods of this type exploit the fact that at the place of the edge, the first derivative of the brightness function (in the direction across the edge) takes its extreme value. The absolute value of derivative describes the strength of the contour at that point and it is called the size of edge. The operators that determine the size of edge are usually called edge operators. The size of edge is usually computed at every pixel of the image. It follows that the most simple edge operators are the derivatives $\partial f / \partial x$ a $\partial f / \partial y$. These operators describe the changes of brightness along the direction of the x a y axis, respectively. It would be possible to use these operators for detecting the edges that are parallel with the coordinate axes. For determining the edges of an arbitrary direction, it is necessary to examine the brightness function in the direction which is perpendicular to the possible edge. Let the vector $\mathbf{n} = (\cos\theta, \sin\theta)$ describes the direction which is perpendicular to the direction of possible edge, and ξ let be the coordinate measured in this direction. For determining whether the edge of the known direction exists at a given point, the derivative in that direction may be used. We have

$$\frac{\partial f}{\partial \xi} = \mathrm{grad}(f) \cdot \mathbf{n} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta \, .$$

The value of $|\partial f / \partial \xi|$ may be regarded as a size of the edge at the point in question. The problem, however, remains that the direction of the edge is not usually known in advance. The ideas we have presented up to now suggest how to solve the problem. For deciding whether or not the edge is present at a given point, the direction in which the change of brightness is the highest is decisive. This direction is determined by the gradient of the image function (brightness function). The edge direction is then perpendicular to the gradient direction. The size of gradient may also be taken as the size of edge.



Determining the edges by the gradient methods

Let $e(x,y)$ be the size of edge, $\varphi(x,y)$ let be the direction of the gradient, and $\psi(x,y)$ let be the direction of edge at the point (x,y). For brevity, we will introduce the notation $f_X(x,y) = \partial f(x,y)/\partial x$, $f_Y(x,y) = \partial f(x,y)/\partial y$. Then we have:

$$e(x,y) = \sqrt{f_x^2(x,y) + f_y^2(x,y)},$$

$$\varphi(x,y) = \arctan\left[\frac{f_y(x,y)}{f_x(x,y)}\right], \qquad \psi(x,y) = \varphi(x,y) + \frac{\pi}{2}.$$

The goal of the practical computation usually is to decide whether or not a point is an edge point. In the simplest case, it is possible to decide between edge and non-edge points on the basis of thresholding. The points where the size of edge is greater than a chosen threshold are considered to be edge points. This method, however, has its drawbacks. The boundary is usually thicker than one image point (intuitively, we usually imagine the boundary of objects as a line whose thickness is one image point). The problems also arise with missing boundary points and, on the other hand, with superfluous boundary points. These problems may be at least reduced by more sophisticated approaches.

Up to now, we have supposed that the image function is a continuous function. This, however, is not the case in practical computation. For discrete image representation, we have to replace the derivatives by differences. We have

$$f_x(x,y) = f(x+1,y) - f(x,y),$$
$$f_y(x,y) = f(x,y+1) - f(x,y).$$

In the past, several methods for computing appeared. We will mention at lest some of them.

Roberts operator: This edge operator is probably the oldest one, but it is mentioned up to these days. The operator may be viewed in such a way that it computes the derivatives in two orthogonal directions (45°, 135°) and then computes the size of the gradient. The prescription is the following one

$$e(x,y) = \sqrt{\left(f(x,y) - f(x+1,y+1)\right)^2 + \left(f(x+1,y) - f(x,y+1)\right)^2}.$$

Prewitt operator: For computing the derivatives and the edge size at a point whose coordinates are *x*, *y*, this operator uses the value of brightness in the neighbouring pixels. Let us denote these values by *A*,*B*,*C*,*D*,*F*,*G*,*H*,*I* respectively as is depicted in the following figure.

A	B	C
D	$f(x,y)$	F
G	H	I

On computing the Prewitt operator

The operator then computes the derivatives by making use of the following formulae that can be interpreted as computing averages of derivatives, which contributes to the noise reduction

$$f_x(x,y) = \frac{1}{6}\left[(C - A) + (F - D) + (I - G)\right],$$

$$f_y(x,y) = \frac{1}{6}\left[(A - G) + (B - H) + (C - I)\right].$$

Sobel operator: This operator resembles to the Prewitt operator. The only difference is that the result is computed by the weighted average instead of the simple average. The weights in particular rows (columns) are 0.25, 0.5, and 0.25 respectively. The formulae than take the form of

$$f_x(x,y) = \frac{1}{4}\left[(C - A) + 2(F - D) + (I - G)\right],$$

$$f_y(x,y) = \frac{1}{4}\left[(A - G) + 2(B - H) + (C - I)\right].$$

Kirsch operator: Another method that can be regarded as a method that is based on the gradient size is the method that was proposed by Kirsch (the method is also known as a Kirsch operator). The value that has the nature of a derivative is computed in eight directions. The biggest value is than taken as the size of edge. Let us denote the values of brightness in image as follows.

A_0	A_1	A_2
A_7	$f(x,y)$	A_3
A_6	A_5	A_4

On computing the Kirsch operator

The size of edge at point whose coordinates are (*x*, *y*) may then be computed according the formula.

$$e(x,y) = \max_{i=0}^{7}[\,|\,5S_i - 3T_i\,|\,],$$

where

$$S_i = A_i + A_{i+1} + A_{i+2}\,,$$
$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7}\,.$$

The index i, for which $e(x,y)$ takes its maximum, determines the direction of edge. Let it be finally pointed out that in case that the values of the indexes in the above formulae should be greater than 7, they are adjusted by carrying out the modulo 8 operation.

Detecting edges by the zero crossing method

From the figure that depicts the derivatives along the direction across the edge, it is clear that, at the place of edge, the second derivative has two extreme values with opposite signs. Since the direction of edge is not usually known in advance, the second derivative should be computed at least in two directions, e.g., in the direction of the x axis and in the direction of the y axis. For this purpose, the Laplace operator may be used. Let us introduce the following notation $f_{xx}(x,y) = \partial^2 f(x,y)/\partial x^2$, $f_{yy}(x,y) = \partial^2 f(x,y)/\partial y^2$. The Laplace operator is then defined by the equation

$$\nabla^2 f(x,y) = f_{xx}(x,y) + f_{yy}(x,y)\,.$$

Since the image function is usually a discrete function, the derivatives must be approximated by differences. The usual formulae are stated below (it is an easy matter to deduce them by repeatedly using the formula for the first difference)

$$f_{xx}(x,y) = f(x-1,y) - 2f(x,y) + f(x+1,y)\,,$$
$$f_{yy}(x,y) = f(x,y-1) - 2f(x,y) + f(x,y+1)\,.$$

By making use of the above formulae, we may easily deduce the formula for computing the Laplace operator. We obtain

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)\,.$$

The above value may be computed by making use of the convolution with the left mask from the following picture. In this context, sometimes also the mask from the right figure is mentioned. It can be explained as a mask that corresponds to computing the sum of the second derivatives computed in four directions, namely, 0°, 45°, 90°, 135°.

0	1	0
1	- 4	1
0	1	0

a)

1	1	1
1	- 8	1
1	1	1

b)

On computing the Laplace operator by making use of convolution

It is obvious that in the border lines and columns, the above masks cannot be used. There are two possible solutions to this problem: (i) the values are computed only in inner pixels in which it is possible, (ii) special masks may be deduced for the pixels on the boundary and at the corners of the picture. The examples are shown in the following figure (again, it is easy to deduce them).

0	1	0
1	- 3	1
0	0	0

a)

0	1	0
0	- 2	1
0	0	0

b)

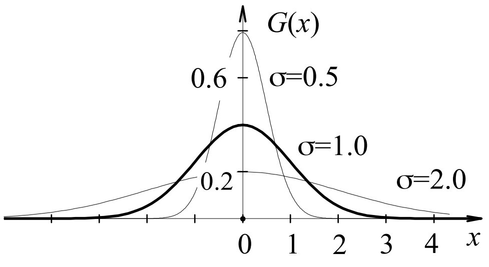
On computing the Laplace operator on the border and at the corner of image.

Let it be recalled that computing the value of the Laplace operator is only the first step of finding the edges. After computing the value of operator, the places of zero crossings must be found, i.e., the places at which the result of the operator takes the zero value between the extreme values with opposite signs. The process of finding these zero crossings may be organised, for example, in lines and in columns separately.

The method based on second derivatives is known as the method that is strongly sensitive to noise. Therefore, a method for noise reduction is usually required prior to edge detection. Usually, the convolution wit the Gaussian function is used at this place. Maar and Hildreth showed that the convolution with the Gaussian function and computing the Laplace operator may be carried out in one step. The two-dimensional Gauss function is defined as follows

$$G(x,y) = G(x)G(y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right).$$

The one dimensional Gaussian is depicted in the following figure (σ influences the width of the function; the width then influences the extent of denoising the image).



The Gaussian function $G(x)$ for various values of σ

The convolution of the image with the Gaussian function may be expressed by the equation

$$\nabla^2\left[G(x,y)\star f(x,y)\right] = \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\int_{-\infty}^{\infty}\int G(x-\xi, y-\eta) f(\xi,\eta)\,\mathrm{d}\xi\mathrm{d}\eta\,.$$

$$= \left[\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)G(x,y)\right]\star f(x,y) = \left[\nabla^2 G(x,y)\right]\star f(x,y)\,.$$

From the above equation, it follows that the convolution with the Gauss function and the subsequent computation of the Laplace operator may be realized in one step as computing the convolution of the original image with the function $\nabla^2 G$. Let us introduce $r = \sqrt{(x^2 + y^2)}$. The function $\nabla^2 G$ then may be written in the form of

$$\nabla^2 G(r) = \left(\frac{r^2 - 2\sigma^2}{2\pi\sigma^6} \right) \exp\left(\frac{-r^2}{2\sigma^2} \right).$$

(The above expression may be derived from the fact that we are computing the Laplacian of the Gaussian function. It takes some time.)

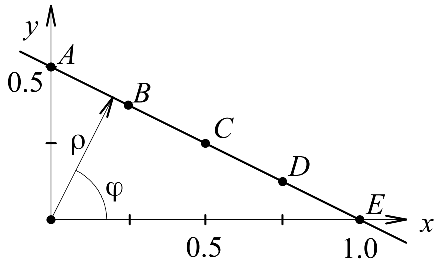
Image Analysis – Lecture No. 2

The Hough transform

Let us say that we have already detected the edge points. The goal is now to decide whether some of them lie on line (on lines). The task may be solved by making use of the Hough transform. Let us consider the equation of line in the form of

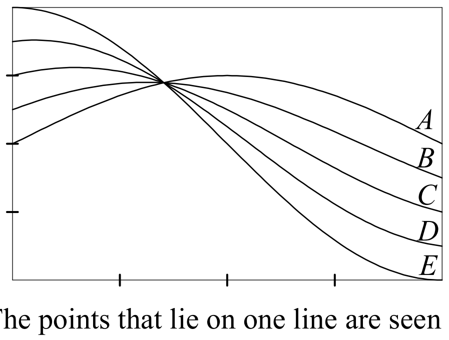
$$\rho = x \cos \varphi + y \sin \varphi \, .$$

For the line that passes through Q whose position is described by the vector (x_Q, y_Q), the following equation holds $\rho = x_Q \cos \varphi + y_Q \sin \varphi$. Let us now consider the space of φ, ρ. In this space, each line passing through Q corresponds to one point. The bundle of lines that pass through Q correspond to the sinusoid (sine curve) $\rho = x_Q \cos \varphi + y_Q \sin \varphi$. Let us now have in image, for example, four points A, B, C, D, E lying on one line (see the following figure).



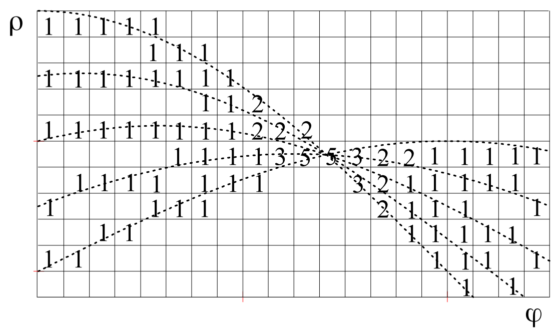
The Hough transform – detecting the points lying on a line.

The bundles of lines that pass through these points correspond to the sine curves A, B, C, D, E in the space φ, ρ. The image of the line on which the points A, B, C, D, E lie is one point in the space φ, ρ in which all the sine curves intersect.

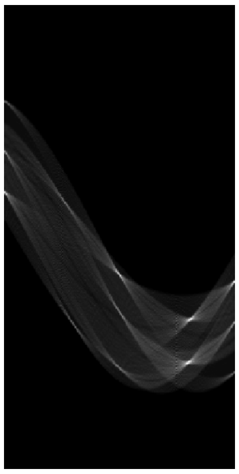


The points that lie on one line are seen as one point in the space of φ,ρ.

The thoughts we have presented up to now lead to the following computational method. The space of φ, ρ is discretised. In this way, we obtain a two-dimensional array of counters (see the following picture). For each edge point in the image, the corresponding sine curve is generated and the corresponding counters are increased. The values of φ, ρ for which the values in the counters are maximal correspond to lines in image.



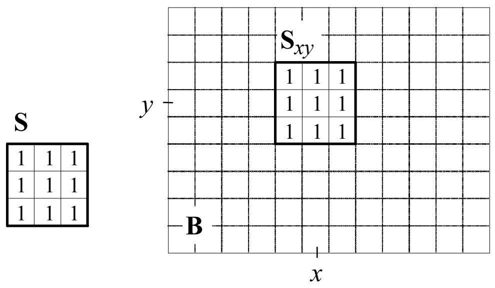
The points lying on one line correspond to a high value of the counter with the corresponding values of φ, ρ.



If we depict the numbers of edge pixels as brightness, we may obtain something like this.

Mathematical morphology

Mathematical morphology in its simplest form is destined for processing binary images. It was introduced by Serra (Serra 82). The theory of mathematical morphology is relatively extensive. We will restrict ourselves only to several elementary examples. Let us denote by **B** the input binary image. Furthermore, we will work with a supplemental binary image that will be denoted by **S**. In mathematical morphology, this image is often called a structural element. The meaning of this supplementary image resembles to the meaning of convolution mask. We will put this image onto different places of **B** and we will use the notation $\mathbf{S}_{xy}$v for the image that arose by moving **S** to the point (x,y) (see the following image).



On the meaning of structural element in morphological operations

The most basic operations in mathematical morphology are called erosion and dilatation. They are defined by the following formulas (**E** and **D** stand for the images that occurs as a result of erosion and dilatation, respectively).

$$\mathbf{E} = \mathbf{B} \otimes \mathbf{S} = \{ x, y | \mathbf{S}_{x,y} \subseteq \mathbf{B} \} ,$$

$$\mathbf{D} = \mathbf{B} \oplus \mathbf{S} = \{ x, y | \mathbf{S}_{x,y} \cap \mathbf{B} \neq \oslash \} .$$

Let us explain the meaning of the first formula (the second formula may be explained in a similar way). By carrying out the erosion of the input image **B** using the structural element **S**, a new image, denoted by **E**, will arise. Let us say that in particular pixels of **B**, there are the values of either 1 or 0. The same also holds for **S**. At the point (x, y), the image **E** has the value of 1 if **B** has the value of 1 at least at those points at which $\mathbf{S}_{xy}$ has the value of 1. Otherwise, **E** has at (x, y) the value of 0. Let it be pointed out that from the formal point of view, the images are here understood as sets of points – those points at which the mentioned binary value is 1.

Further morphological operations that are frequently used are opening and closure. The opening is defined as erosion that is followed by dilatation. The closure is, on the other hand, the dilatation followed by erosion. The corresponding formulas are the following ones

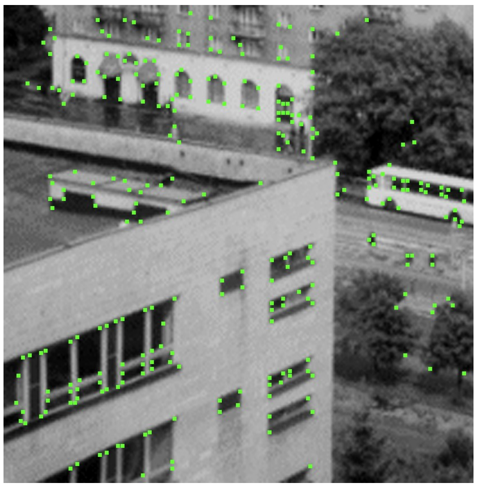
$$\mathbf{B} \circ \mathbf{S} = (\mathbf{B} \otimes \mathbf{S}) \oplus \mathbf{S} ,$$

$$\mathbf{B} \bullet \mathbf{S} = (\mathbf{B} \oplus \mathbf{S}) \otimes \mathbf{S} .$$

From the above definition, it should be clear that the opening eliminates small and very thin objects. It also splits the objects at places where they are very thin. The closure, on the contrary, fills small or thin holes (e.g., hair cracks) in objects and connects together the objects that lie very close one to another. Alternatively to the formulas stated above, also a higher number of erosions and dilatations may be used.

Detecting corners

Sometimes, further processing of images may be based on finding the corners (feature points). The use of corners may be advantageous for the following reasons: (i) the corners are usually abundantly present in images (both in the images of scenes that were created by humans and in the images of natural scenes), (2) in image, the position of corner may be unambiguously determined, (iii) the corners in images usually correspond to vertices of real three-dimensional objects. These vertices are of interest, e.g., in scene reconstruction (creating a three-dimensional model of a scene from two or more images). We will mention at least two famous corner detectors.



The Kitchen and Rosenfeld corner detector computes the product of the gradient and the Gauss curvature of the surface that is defined by the brightness function. The idea is that the corners are at the points at which the value of the Gaussian curvature is big (extreme). The formula for computing the curvature is the following one (we state it without deducing it; I stands for the image function, I_x for its derivative in the x direction etc)

$$k = \frac{I_{xx} I_y^2 + I_{yy} I_x^2 - 2 I_{xy} I_x I_y}{I_x^2 + I_y^2} .$$

The detector proposed by Harris is one of the most frequently used. It is regarded as very stable and reliable. In this detector, the following value is computed at each point

$$\mathbf{M}(x, y) = \begin{bmatrix} \left\langle (I_x)^2 \right\rangle & \left\langle I_x I_y \right\rangle \\ \left\langle I_x I_y \right\rangle & \left\langle (I_y)^2 \right\rangle \end{bmatrix} .$$

The notation $\langle\rangle$ means the convolution with the Gaussian function, i.e.,

$$\left\langle \psi(x, y) \right\rangle = \psi(x, y) * \left[G(x) G(y) \right] ,$$

where * stands for convolution and $G(u)$ is the Gaussian function

$$G(u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{u^2}{2\sigma^2} \right).$$

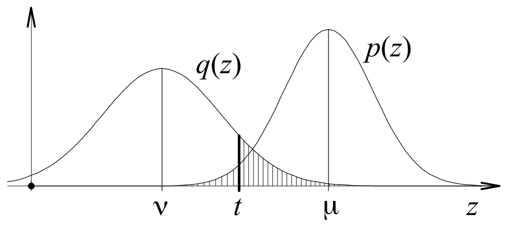
The choice of the value of σ influences the sensitivity of the detector. The final decision whether or not a point (x, y) is a corner point is done on the basis of the following expression

$$\text{cor}(x, y) = \det\left(\mathbf{M}(x, y)\right) - 0.04 \cdot \text{trace}^2\left(\mathbf{M}(x, y)\right) .$$

The points at which the above value is extreme and big enough may be regarded as corner points.

Finding the value of threshold for thresholding

In this paragraph, we will discuss the problem of finding the value for thresholding. We have discussed the thresholding itself earlier in this text. For determining the value of threshold, we will use the method of minimising the probability of error. We suppose that the probability density for brightness of objects is $p(z)$ and the probability density for the brightness of the background is $q(z)$ as is depicted in the following picture.



On determining the threshold by making use of error minimisation

Furthermore, we will assume that in the image, the ratio (number of object pixels) / (total number of pixels) is θ. Similarly, for the background, we have the corresponding ratio 1-θ. Suppose that the threshold is *t*. Let us denote by *P*(*t*) the probability that the object pixel will be incorrectly classified as a pixel of the background. Similarly 1 – *Q*(*t*) is the probability of the event that the background pixel will be classified as an object pixel. For probabilities *P*(*t*), *Q*(*t*), we have

$$P(t) = \int_{-\infty}^{t} p(z)\,\mathrm{d}z \ , \quad Q(t) = \int_{-\infty}^{t} q(z)\,\mathrm{d}z \ .$$

We are now ready to express the error, denoted by ε, that arises during the thresholding. We should also consider how the pixels of object and the pixels of image are frequent in images we work with. It can be done by taking the values of θ and 1-θ as weights as follows

$$\varepsilon = \theta P(t) + (1 - \theta)\left[1 - Q(t)\right] \ .$$

For finding the minimum, we will compute the derivative of ε, and we will set this derivative equal to 0. We have

$$\frac{\partial \varepsilon}{\partial t} = \theta \frac{\partial P(t)}{\partial t} - (1 - \theta)\frac{\partial Q(t)}{\partial t} = \theta p(t) - (1 - \theta) q(t) = 0 \ .$$

It follows that $(1 - \theta) q(t) = \theta p(t)$.

It may happen that the probability densities have normal distribution, i.e.,

$$p(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(t - \mu)^2}{2\sigma^2}\right] , \quad q(t) = \frac{1}{\tau\sqrt{2\pi}} \exp\left[\frac{-(t - \nu)^2}{2\tau^2}\right] \ .$$

In that case, by logarithming the previous equation, we obtain

$$\ln(1 - \theta) - \ln \tau - \frac{(t - \nu)^2}{2\tau^2} = \ln \theta - \ln \sigma - \frac{(t - \mu)^2}{2\sigma^2} \ .$$

After rewriting, we have $\tau^2(t - \mu)^2 - \sigma^2(t - \nu)^2 = 2\sigma^2\tau^2 \ln \frac{\tau\theta}{\sigma(1 - \theta)}$.

The above expression is a quadratic equation for *t*. For solving that equation, the values of μ, σ, ν, τ, θ must be known. For the special case θ = ½, σ = τ, we can easily obtain *t* = (μ + ν) / 2, which is not surprising.