Introduction to Image Processing

Matheus Zaia

Image processing aims to extract the maximum amount of information from the data obtained in a particular experiment, being problem oriented, meaning that there is no such thing as a universal solution, only a case to case solution.

In general, an image processing workflow follows like this: data acquisition, pre-processing, segmentation, and data analysis. Data acquisition involves transforming the signals obtained by the detector of a certain equipment into digital signals, creating a digital image. Pre-processing involves applying filters to the raw image, performing image enhancement to facilitate subsequent steps. Segmentation is the process that separates the objects contained in the image, for example, in an image of a cell, we can separate it into membrane, nucleus, and organelles. Finally, analysis is the quantification performed on the final image obtained through segmentation, such as calculating area, perimeter, average intensity, among other characteristics of the obtained objects.

Digital Image Basics

A digital image can be defined as a function of two variables, $f(x, y)$, where x and y take discrete values. The value of the function at a point (x, y) is represented by its intensity, always a positive value. Basically, an image is a matrix A_{xy}, with its elements $a_{ij} \in A$ representing the pixels of the image. The variables x and y constitute the spatial domain of the image, representing spatial coordinates. Since the image is represented by a matrix, the values of x increase downward, while y increase to the right.

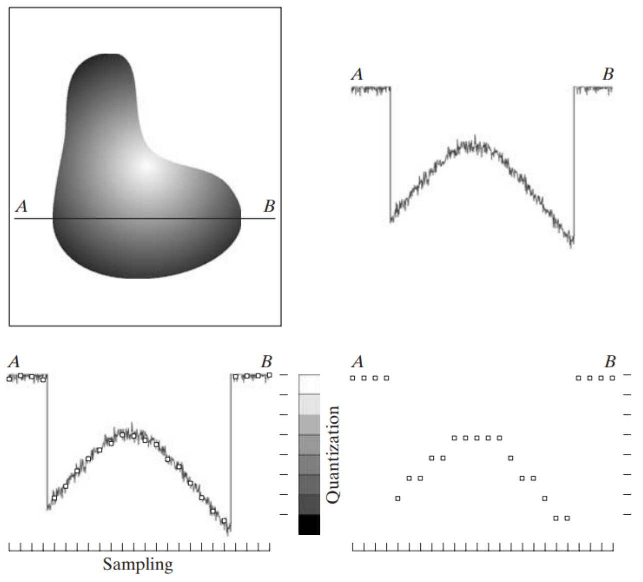
The function f consists of two components, namely the signal that interacts with the sample, $i(x, y)$, and the signal that comes back after the interaction, $r(x, y)$. The component i can vary between $0 < i(x, y) < \infty$, on the other hand, the component r varies from $0 < r < 1$, where 1 is the maximum sensor reading and 0 is the minimum. Both of them on the types of objects in the sample and the physical phenomena being observed.

Sampling and Quantization

To transform the data obtained into digital images, it is necessary to determine the number of points the image will have, and this step is called sampling. These points are discrete and will determine the size of the matrix that will later be transformed into an image.

The quantization process is the step that will define the intensity intervals that each point established by sampling will assume. These points are obtained after defining the intervals that will classify the continuous values obtained by the detector into discrete values. As a standard for grayscale images, intervals from 0 to $L-1$ are used, where the value 0 is represented by the color black, and $L-1$ is represented by the color white.

Figure 1: Sampling and quantization



Pixel Depth and Resolution

Pixel depth is the number of discrete values a pixel can take, with these values being integers in the interval $[0, L-1]$, in the case of a gray scale image. As information is stored in bits, we have $L = 2^k$, and an image with 2^k bits of depth is referred to as a k-bit image. When a pixel reaches the maximum value it can take, it is called a saturated pixel, indicating that the obtained data may not be reliable, as it is impossible to determine whether the pixel's value is precisely $L-1$ or a higher value. Another important definition is the image contrast, which is the difference between the highest and lowest intensity values in the image.

The intensity resolution of an image depends on its pixel depth: the higher the pixel depth, the greater the image definition. However, it is important to note that the human ability to distinguish color tones is limited. Therefore, it is not advantageous to use an unnecessarily large pixel depth, because it will lead to unnecessary memory allocation.

It is also possible to consider the spatial resolution of the image, which can be quantified by the number of pixels per a certain size. The smaller the pixel size in the image, the higher the spatial resolution of the image.

Filters

Filters are mathematical operations performed on the image to enhance certain features. There are various classes of operations, like pointwise, spatial, morphological and frequency operations. Each one of them has their own applications, reinforcing the problem oriented nature of image processing.

Pointwise Filters

Pointwise filters perform pixel by pixel transformations, without considering their neighborhood. A pointwise transformation can be described as:

$$g(x, y) = T[f(x, y)]$$

Where f is the original image, T is the transformation used, and g is the final image.

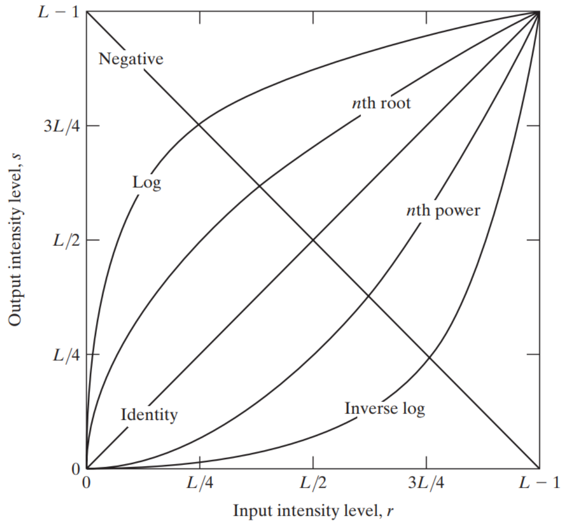
Negative Filter

The negative filter inverts the values with respect to their grayscale tone, meaning it assigns the complementary value with respect to the maximum intensity allowed by the pixel as the new intensity value. The transformation is given by:

$$s = L - 1 - r$$

Where s is the intensity output and r is the intensity input of each pixel.

Figure 2: Pointwise transformations



Logarithmic Filter

Logarithmic transformations follow the form:

$$s = c\log(1 + r)$$

Where c is an arbitrary constant. This filter enhances the difference between low intensity values and decreases the difference between higher intensities. To obtain the inverse result, we use the inverse logarithmic transformation, which, in turn, accentuates differences between higher intensities and decreases differences between lower intensities.

Gamma Filter

Gamma filters are of the following form:

$$s = cr^\gamma$$

These transformations have a similar effect to logarithmic transformations, but now it is easier to observe the two possible classes of transformations. With $\gamma > 1$, we spread the higher intensities, while the lower values are compressed, whereas with $\gamma < 1$, we have the opposite effect.

Spatial Filters

Spatial filters take into account the neighborhood of each pixel and consist of a kernel and a predefined operation. The algorithm of a spatial filter follows the following logic: iterate over all the pixels of our image, overlay the central element of the kernel with each pixel, perform a predefined operation, and finally assign the result of this operation as the intensity for the pixel of the filtered image.

Correlation and Convolution

Spatial filters are based on the processes of correlation and convolution. The correlation operation is analogous to convolution, except with the 180° rotation of the kernel used.

Convolution is based on the process of using a kernel, a matrix smaller than the original image, which will scan each pixel of the image, multiply each item of the overlay of the kernel with the image, sum the results, and assign this result as the new value of the pixel that overlaps with the center of the kernel.

Before convolution is performed, the kernel is rotated by 180°. The reason is that when the kernel is not rotated, the described process applied to a discrete impulse generates the matrix rotated by 180°, which represents the correlation process. With the kernel inverted before scanning the image, the result obtained is the original kernel.

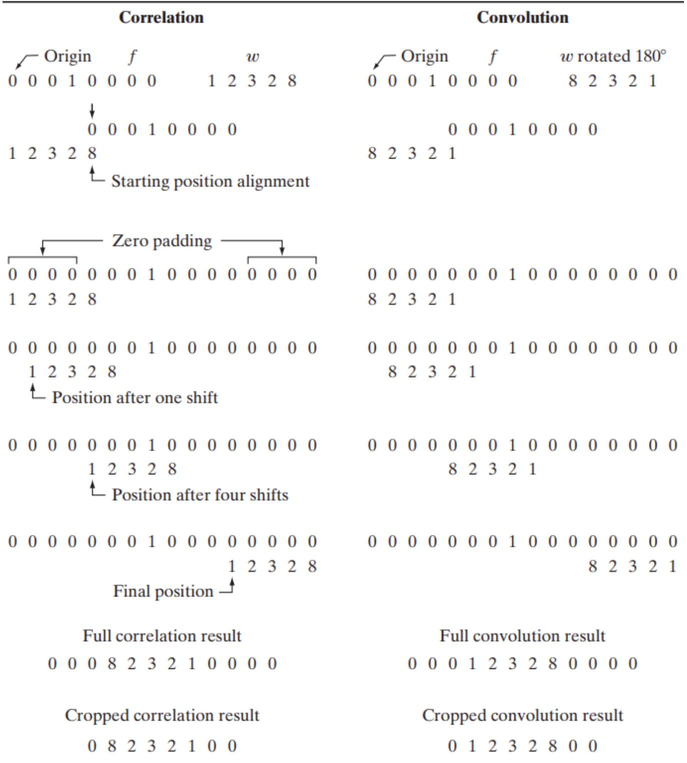
Formally, given a kernel w_{3x3} and a region of the image f, the convolution at point (x, y) will be given by:

$$w(x,y) * f(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) f(x-s, y-t)$$

The correlation operation uses the signals of s and t inverted as we traverse the pixels of the region f. It is also important to note that if the kernel is symmetric, correlation and convolution operations result in the same process.

With an understanding of the convolution process, it is possible to construct kernels that use linear or non-linear operations to enhance the image in question. To simplify the notation, let's consider only 3x3 kernels.

Figure 3: Correlation and convolution



Mean Filter

To create a mean filter, the kernel must have coefficients equal to $\frac{1}{9}$.

$$w = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

This filter is used to blur the image, as the new intensity of the pixel becomes closer to its neighbors.

This relationship can be altered by using a weighted average, assigning certain weights to each position of the kernel and then dividing each element by the sum of all weights. This way, each pixel can have a different weight in the calculation of the average.

Gaussian Filter

To construct the Gaussian filter kernel, we use a function of the form $g(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ at each coordinate of the kernel. Using $\sigma = 1$, the following matrix is obtained:

$$w = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

The Gaussian filter has the effect of blurring the image, and the larger the value of the standard deviation, σ, the greater this effect. This filter is classified as a low pass filter, as it reduces the frequency of high valued pixels.

Non-Linear Statistical Filters

Gaussian and mean filters are examples of linear filters. However, nonlinear operations are also applied to the image, such as median, maximum, and minimum filters, among others.

The median filter is widely used to remove salt and pepper noise, for example. Since this type of noise includes the presence of pixels with maximum and minimum values scattered throughout the image, the median filter removes these pixels in the convolution process, leaving the image more homogeneous.

Sharpening Filters

Unlike blurring filters, sharpening filters aim to highlight the transition between different intensities. This results in the filtered image having a focus on edges and lines that delineate objects, with less definition in regions of constant intensity.

To understand enhancement filters, it is necessary to understand the derivative of an image. Since the values of a pixel are discrete, the derivative at a point (x, y) is defined by:

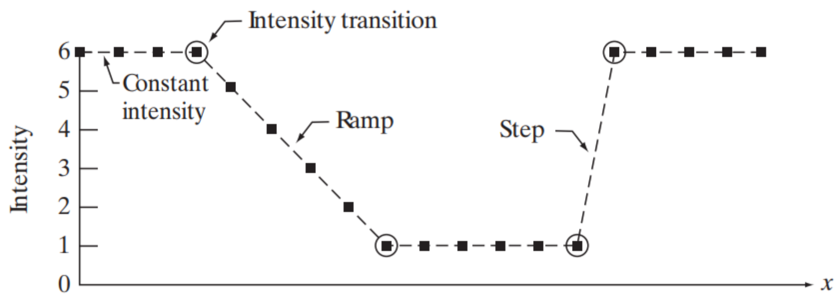
$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

As images are two-dimensional objects, partial derivative notation will be used. The definition of a derivative needs to meet some requirements due to the discrete nature of the values being treated: in regions of constant intensity, the value must be zero; in ramps and intensity steps, the value must be different from zero.

We often use the second derivative of the image, following the definition of the first order derivative.

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

Figure 4: First and second order derivatives



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

As derivatives require the subsequent point of the analyzed point, there may be issues with the edges of the image during the implementation of algorithms, and this can be corrected with zero padding, the addition of zeros at the edges of the image.

Laplacian Filter

The Laplacian of a function at a point (x, y) is given by the sum of the second-order simple derivatives at that point, that is:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Considering only horizontal and vertical derivatives, this filter is isotropic at 90°, meaning that rotations of 90° do not alter the result. To capture even more details, it is possible to add mixed partial derivatives to the filter, making it isotropic at 45°. Below are the two possible kernels used for the filter, the first isotropic at 90°, the second at 45°:

$$w_{90} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad w_{45} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

As mentioned earlier, the Laplacian filter highlights intensity contrasts, resulting in an image with well-contrasted lines and edges, but with a gray background and no details. The filter is usually applied as follows:

$$g(x, y) = f(x, y) - c\nabla^2 f(x, y)$$

With c being an arbitrary constant.

Unsharp Mask and Highboost

The unsharp mask filter uses a blurred image of the original image to increase contrast. The mask of the filter, g_m, is the difference between the original image, f, and its blurred version, f^*:

$$g_m(x, y) = f(x, y) - f^*(x, y)$$

Given the mask, we just add it to the original image:

$$g(x, y) = f(x, y) + k \cdot g_m(x, y)$$

When the scalar k is equal to 1, the filter is called unsharp mask, and when $k > 1$, the filter is called highboost, creating a bigger contrast.

Gradient Filter

The gradient of a function, ∇f, is the vector containing its partial derivatives.

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

Thus, the gradient of an image is the image containing, as pixel intensity, the value of the gradient for each point (x, y) of the original image. To perform these calculations, there are options of kernels that compute the gradient of the image, being some of them: the Robert operators, the Sobel operators, the Prewitt operators, and the Scharr operators.

The Robert kernels consider a diagonal displacement around the pixel in question, and they are represented below:

$$w = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \qquad w = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Since 2x2 kernels do not have a central element, there are other kernels that compute the gradient using 3x3 matrices, although the 2x2 matrices can be used as well. Below are the Sobel kernels:

$$w_h = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad w_v = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Below are the Prewitt kernels:

$$w_h = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad w_v = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

And lastly, the Scharr kernels:

$$w_h = \begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{pmatrix} \qquad w_v = \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix}$$

All these filters enhance the edges of the objects present in the image and darken regions with similar intensity. The indices h refer to filters that capture horizontal edges, while the indices v refer to vertical edges. It is also important to note that for ease of notation, the filters are already in their convolution form, without the need for rotation by 90° (that is the case for most kernels you might see).

Morphological Filters

Morphological filters use the area of mathematical morphology, based on sets, to perform their transformations, differing from the operation of pointwise and spatial filters.

Math Morphology Foundations

The theory of mathematical morphology is based on sets and their operations, and thus, we can address various image processing problems. In the case of images, the elements of a set are tuples, which represent the coordinates of the pixel, along with extra components that provide other information about that pixel, such as color or grayscale value.

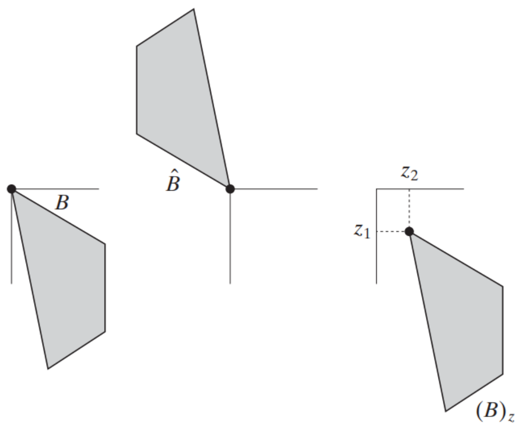
Entering into some types of set operations, the reflection of a set is defined as:

$$\hat{B} = \{x | x = -b,\ b \in B\}$$

The translation of a set B is defined as:

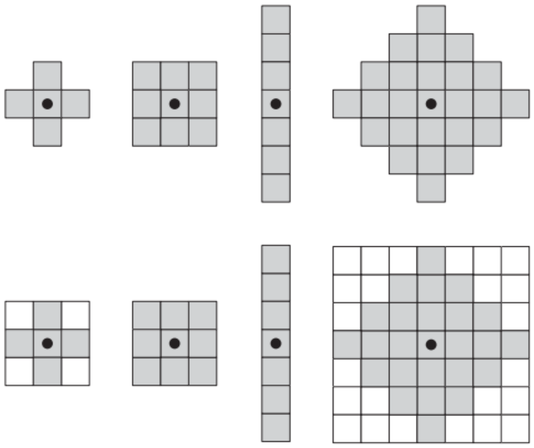
$$B_z = \{x | x = b + z,\ b \in B\}$$

Figure 5: Reflection and translation of a set B



Another important concept are the structural elements (SE), which are small sets used for morphological operations. SEs are always rectangular and have an origin, which is usually located at the center point or symmetry point of the element. It is important to note that the choice of the SE is going to impact the shape of the objects in the final image.

Figure 6: Structural elements (SE)



Morphological operations are usually done in binary images, although they can be applied in grayscale images as well.

Erosion

The erosion of a set A by a set B, with $A, B \in Z^2$ is given by:

$$A \ominus B = \{z |\ B_z \subseteq A\}$$

The intuition behind erosion is that the set B, which is representing a SE, is going to be translated through the set A, and every time the non-zero pixels of B are completely over non-zero values of set A, the pixel z assumes a value of 1, and otherwise, a value of zero.

Erosion shrinks the objects in the image, being often used to remove connections that are not desired in the image.

Dilation

The dilation of set A by set B is defined as:

$$A \oplus B = \{z |\ \hat{B}_z \cap A \neq \emptyset\}$$

The intuition behind dilation is similar to erosion, but this time, if at least one non-zero pixel of the SE is over a non-zero pixel of the set A, the pixel assumes a value of 1, and zero otherwise.

This operation increases the size of the objects, being sort of an inverse operation with respect to erosion, intuitively speaking.

Opening

The opening operation is given by an erosion, followed by a dilation:

$$A \circ B = (A \ominus B) \oplus B$$

The operation leaves the edges smoother, and remove small objects.

Closing

Closing does the opposite of opening, performing a dilation followed by a erosion:

$$A \bullet B = (A \oplus B) \ominus B$$

The effect here is that small holes are filled, instead of removing small objects, also leaving edges smoother.

Boundary Extraction

We can extract the boundary, $\beta(A)$, of a binary image by subtracting its erosion image.

$$\beta(A) = A - (A \ominus B)$$

Segmentation

Segmentation is one of the last parts of the pipeline of image treatment, being responsible for separating objects in the image. We can do this process in various ways and approaches, but mainly: separating the objects from the background, creating a binary image, or; separating each object with its own label.

Threshold

Thresholding is a technique that separates objects from the background of the image, creating a binary image. In the global threshold, we create a threshold value for the whole image, and for local threshold, the values are determined locally.

Otsu's Threshold

The Otsu method for global threshold searches for the value T that minimizes the intraclass variance, defined as the weighted sum of the class variances. Assuming we have two classes, the value of t is obtained by minimizing the following equation:

$$\min_t \left[w_0(t)\sigma_0^2 + w_1(t)\sigma_1^2\right]$$

Where t is an intensity value, and w_0 and w_1 are the sum of the probabilities of each class, given by:

$$w_0(t) = \sum_{i=0}^{t-1} p(i) \quad \text{and} \quad w_1(t) = \sum_{i=t}^{L-1} p(i)$$

The value of t can be achieved iteratively, calculating the value of the expression above for each $t \in [0, L-1]$.

Local Threshold

Local thresholding is a technique used for images with unequal illumination, not letting us separate the objects from the background with a single global value.

For that, we can use a local kernel, generating a threshold value for each pixel by applying functions like the mean, while translating the kernel through the image. Another option used is the average of the maximum and the minimum in the neighborhood, given by:

$$t(x, y) = \frac{\max(w) - \min(w)}{2}$$

Where $t_{(x,y)}$ is the threshold value for the pixel (x, y), given the neighborhood defined by the kernel w.

Sauvola's Threshold

The Sauvola method for calculating the local threshold is given by the following formula:

$$t(x,y) = \mu(x,y)\left(1 + k\left(\frac{\sigma(x,y)}{R} - 1\right)\right)$$

Where k and R are scalars, and μ and σ are the mean and standard deviation, respectively, of the neighborhood of the pixel (x, y). The higher the value of k, the lower the threshold from the local mean.

Watershed

Metaphorically, the watershed functions like the flooding of several hydrographic basins. Analyzing the image in a three-dimensional manner, it presents various local minima points, which represent the different hydrographic basins. Similarly, the other pixels may belong to a certain basin, meaning that by placing a drop of water at that point, it would fall into a single basin, or they may be at basin boundary locations. The main objective of this algorithm is to find the boundaries between the hydrographic basins.

To find the local minima, we apply a distance transform in the thresholded image, where each pixel is going to assume a value proportional to its distance from the nearest background pixel. Generally, we perform a Euclidean Distance Transform (EDT).

After defining the local maxima points of the image, we apply the negative filter, turning them into local minima points. They will be the seeds of the algorithm, these basins start to be flooded. When the water from one basin encounters that of another, the algorithm defines a boundary between these two basins, defining the separation of two objects. Once all boundaries are established, each object is assigned a label, or name, allowing for the identification of each object individually.

References

[1] *Digital image processing*, GONZALES, Rafael C.; WINTZ, Paul.