Image Analysis - TP2 - Local Filtering and Histograms

Jander Nascimento, Raquel Oliveira

March 12, 2011

1 Filtering

1.1 Binomial

Binomial filter uses Pascal's triangle to create a filter. This filter has the blurry effect with some differences in preserving objects in the image.

One example of a 3x3 Binomial filter can be seen in the Figure 1. The values for this kernel can be obtained from binomial coefficient definition, presented in Equation 1.

$$\frac{n!}{(n-k)!k!} = \binom{n}{k} \tag{1}$$

1	2	1
2	4	2
1	2	1

Figure 1: 3 x 3 Binomial filter

This kernel is used to remove the noise in image, but it do not preserve the edges of the image.

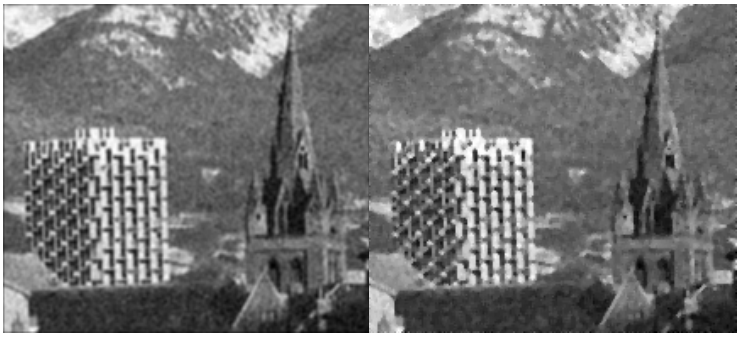
1.2 Median

With a purpose of noise reduction, in certain situations Median Filter can preserve the edges while reduce the noise of the image. For this reason this kind of filter is regularly used as a pre-treatment for edge detection.

The Median filtering differs from others filters in the way it is applied to the image. While other filters are defined by kernels as the aproximation of its derivation, the median filter depends on each sector of the image analyzed to calculate the new pixel intensities.

Median kernel copies a certain sector of the image analyzed and change the value of the central pixel in the original image. The central pixel in the original image receives the median value of the kernel. It is possible to do it by simply sorting the content of the kernel and assign the value in the middle (central position of the kernel) to the original image.

1.3 Binomial *versus* Median



(a) Binomial (b) Median

Figure 2: Removing noise

In the Figure 2 we applied Binomial and Median filter separately. Median preserves the edges better than the Binomial, although the Binomial preserves some central details in the image that the Binomial do not. This behavior can be seen in the window at the chapel, in the Binomial is possible to see the borders of the window more clearly, while in the Median this detail is not displayed at all.

2 Histogram

The histogram shows the color distribution of a certain image regarding to its color intensity. The histogram is represented in a Cartesian, where x axis is the color intensity and y represents the number of pixels that have such intensity.

In this practical work, we wrote a function in c that computes the intensity histogram. The values are stored in a array of 255 positions. We do not display the histogram as a graph. Just its values are displayed.

2.1 Stretching

The histogram stretching is a technique by which the color histogram is used to evaluate and possibly change the color intensity range. This enhances the detail

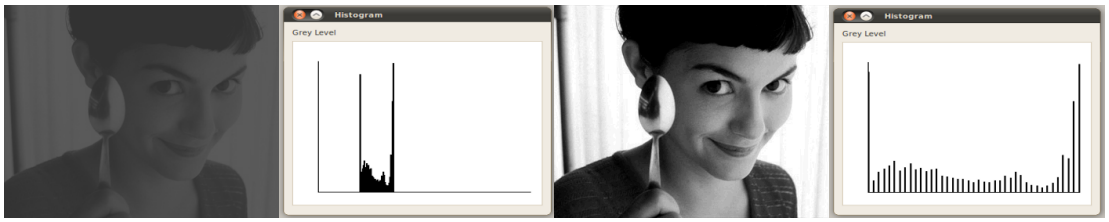
level of some images that might appear too dark or too bright. This is done by spreading the pixels to use the entire color intensity.

For each pixel of the image, we calculate its new intensity based on the stretching formula:

$$y[n] = new_{min} + \frac{new_{max} - new_{min}}{current_{max} - current_{min}} * (x[n] - current_{min}) \quad (2)$$

where:

- n is the pixel
- new_{min} is 0;
- new_{max} is 255;
- $current_{max}$ is the maximum intensity of the color that is used in the image
- $current_{min}$ is the minimum intensity of the color that is used in the image
- *x[n]* is the current intensity of the pixel.



(a) Original Image (b) Original Histogram (c) Stretched Image (d) Stretched Histogram

Figure 3: Stretching Transformation

2.2 Equalization

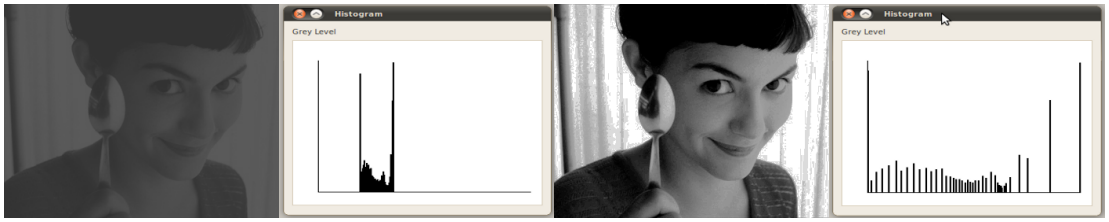
Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast.

Based on the histrogram of the image, we compute the cumulative distribution functions, which is the values of the histogram in a cumulative way. For each pixel of the image, we calculate its new intensity based on the equalization formula:

$$h[v] = round\left(\frac{cdf(v) - cdf_{min}}{(M * N) - cdf_{min}} * (L - 1)\right) \quad (3)$$

where:

- v is the current intensity of the pixel
- *cdf(v)* is the value of such intensity in the array that stores the accumulated values of the histogram
- cdf_{min} is the minimum intensity value of the color
- M is the height of the image
- N is the width of the image
- L is the number of color levels used (in most cases, 256)



(a) Original Image (b) Original Histogram (c) Equalized Image (d) Equalized Histogram

Figure 4: Equalization Transformation

2.3 How to run?

Steps to compile the application:

- svn checkout https://jfimageanalysis.googlecode.com/svn/trunk/TP2/ #download source code
- make #compiles the code

As an input image only **pgm plaintext** files are accepted (P2).
Examples of usage:

- Binomial filter to an image:

  ./imagetransform -f binomial -i image_in.pgm > image_out.pgm

- Median filter with kernel size s, applied n times:

  ./imagetransform -f median -i image_in.pgm -s 3 -n 5 > image_out.pgm

- Equalize

  ./imagetransform -i image_in.pgm -e > image_out.pgm

You can always type *./imagetransform –help* to check for more options.