Image Enhancement with Matlab Algorithms

Julián Calderón González
Òscar Daniel Carmona Salazar

Blekinge Institute of Technology
Department of Applied Signal Processing
SE-371 79 Karlskrona Sweden

Contact Information:
Authors:
Julián Calderón González
jucb14@student.bth.se

Òscar Daniel Carmona Salazar
osca14@student.bth.se

University examiner:
Sven Johansson
Department of Applied Signal Processing

University supervisor:
Ingvar Claesson
Department of Applied Signal Processing

Blekinge Institute of Technology
Department of Applied Signal Processing
SE-371 79 Karlskrona, Sweden

Internet: www.bth.se
Phone: +46 455 38 50 00
Fax: +46 455 38 50 57

Abstract

Nowadays, a lot of applications use digital images. For example in face recognition to detect and tag persons in photograph, for security control, and a lot of applications that can be found in smart cities, as speed control in roads or highways, cameras in traffic lights to detect drivers ignoring red light. Also in medicine digital images are used, such as x-ray, scanners, etc.). These applications depend on the quality of the image obtained. A good camera is expensive, and the image obtained depends also in external factor as light.

To make these applications work properly, image enhancement is as important as, for example, a good face detection algorithm. Image enhancement also can be used in normal photograph, for pictures done in bad light conditions, or just to improve the contrast of an image. There are some applications for smartphones that allow users apply filters or change the bright, colour or contrast on the pictures.

This project compares four different techniques to use in image enhancement. After applying one of these techniques to an image, it will use better the whole available dynamic range. Some of the algorithms are designed for greyscale images and others for colour images. It is used Matlab software to develop and present the final results. These algorithms are Successive Means Quantization Transform (SMQT), Histogram Equalization, using Matlab function and own function, and V transform.

Finally, as conclusions, we can prove that Histogram equalization algorithm is the simplest of all, it has a wide variability of grey levels and it is not suitable for colour images. V transform algorithm is a good option for colour images. The algorithm is linear and requires low computational power. SMQT algorithm is non-linear, insensitive to gain and bias and it can extract structure of the data.

CONTENTS

INDEX OF FIGURES

1 INTRODUCTION

The history of digital image processing is closely linked to development of computers. Digital images require so much storage and computational power, it is necessary to use computers and technologies that match this requirement.

In the 60s and 70s, in parallel with the development of space applications, techniques of digital image processing also began its development in different fields such as medicine, biology, geology or astronomy. Today, due to technological advances in recent years, digital image processing has become a routine task essential for solving problems in numerous applications and devices, so to obtain optimal solutions, the user just needs to understand the problem and know how to apply the tools available in the market. [1]

Overcome the technological barrier, challenge now is to find the documentation that allows new users to understand the operation of programs and techniques involved in digital image processing.

1.1 Context and Motivation

We live in an era in which all forms of information are undergoing a process of digitization. The images, of course, could not escape this process. Photography, film, television, graphic design and even industrial design produce thousands of digital images that are stored on a physical medium, sent by electronic means of transmission, presented on a screen, in any device or printed on paper. [2]

It is necessary a process with aim of reduce the size or an enhancement of the quality of digital images for its transmission or storage, which is called digital image processing. The digital image processing is the set of techniques applied to digital images in order to improve quality or facilitate the search for information. [3]

1.2 Aim of the Project

This project compares four different techniques to use in image enhancement.

- Successive means quantization transform (SMQT) (See section 3.1).
- Histogram Equalization (own function and Matlab function) (See section 3.2).
- V transform; a function about enhancement through rgb2hsv conversion (See section 3.3).

These algorithms are programmed in Matlab.

2 DIGITAL IMAGE PROCESSING

2.1 General Concepts

2.1.1 Light and visible spectrum

If a beam of white light passes through a glass prism, the human can see that the beam of light is broken and the six colors of the spectrum appear: red, orange, yellow, green, blue and violet (figure 1). This is known as refraction and was discovered by Sir Isaac Newton in 1666. In this way, you can understand that white light, existing everywhere, is composed of a spectrum of colors and collided with a body, it absorbs any of these components and reflects others. The reflected colors are what our eyes can perceive. [4]

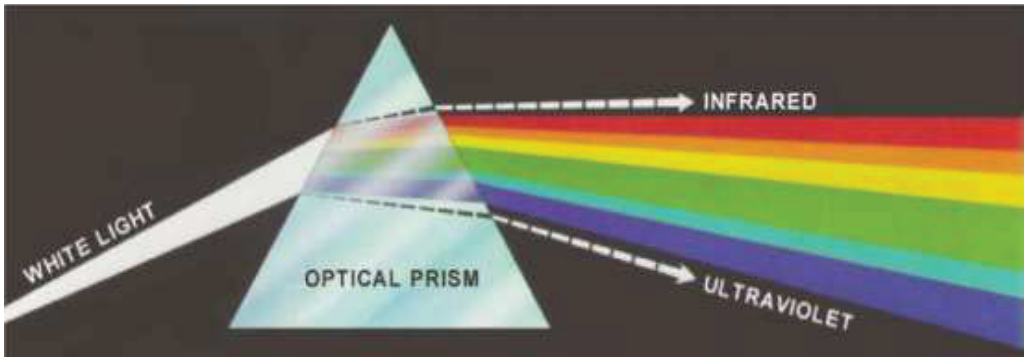

Figure 1. Color spectrum seen by passing White light through a prism

Basically, the colors that human perceive depend on the nature of the light reflected from the object. So the visible range by human eyes can be seen (figure 2) as a little part within the electromagnetic spectrum and include wavelengths from 380 nm to 780 nm. The human eye perceives light from each of these wavelengths as a different color. [4]

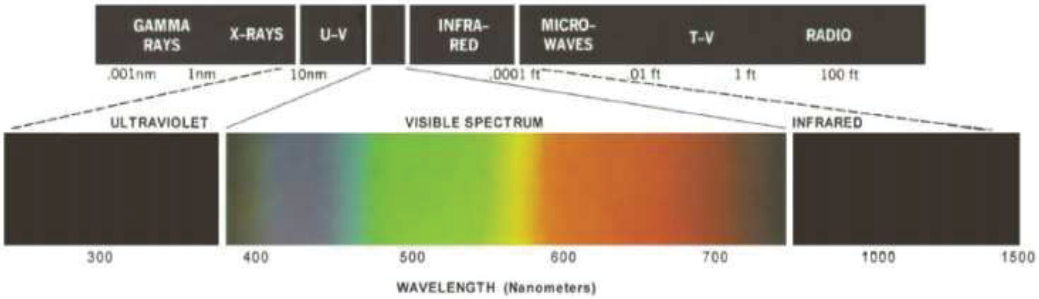

Figure 2. Wavelengths comprising the visible range of the electromagnetic spectrum

Primary color is a color that cannot be obtained by mixing any other one. This is an idealized model, based on the biological response of the receptor cells of the human eye (cones) in the presence of certain frequencies of light and noise, and is dependent on the subjective perception of the human brain. Mixing two primary colors gives rise to a secondary color.

The theories of traditional and modern color disagree on which are the primary colors. The modern color theory distinguishes between light and pigment colors (figure 3). [5]

- Light primary colors (RGB model): Red, green and blue.
- Primary pigment colors (CMY Model): Cyan, magenta and yellow.
- Traditional primary colors (RYB Model): Red, yellow and blue. This model is the precursor CMY model. It is considered obsolete by science and industry.

It is called secondary when a color is obtained by mixing two primary colors and which in turn is complementary color of a third primary color, which is not involved in its preparation. [5]

- Secondary colors light (RGB model) and Cyan, magenta yellow
- Secondary colors pigment (CMY Model): Orange, green and violet.

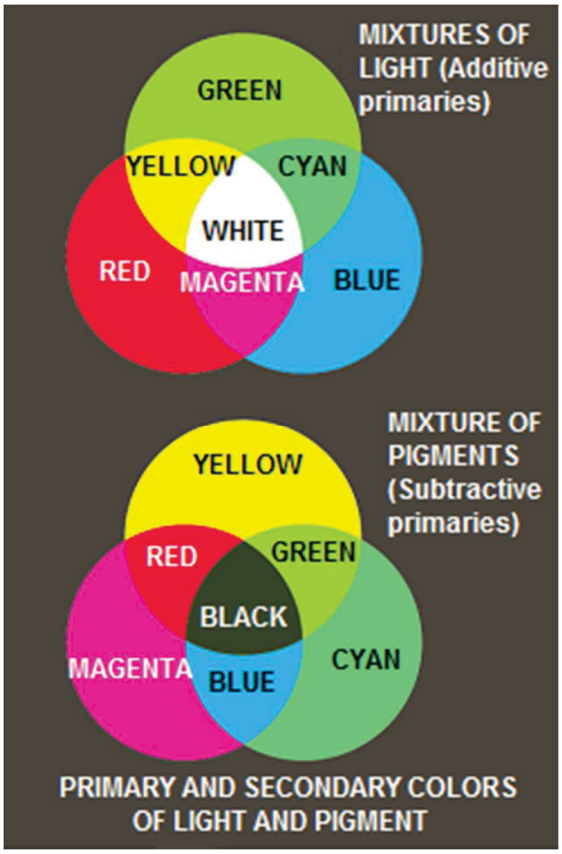

Figure 3. Primary and secondary colors of light and pigments

2.1.2 Digital image

The term image refers to a two dimensional function of light intensity *f (x, y)* where *x* and *y* denote the spatial coordinates and the value of *f* at any point *(x, y)* is proportional to the intensity of the image at that point. A digital image can be written as a matrix whose row and column indices identify a point in the image and whose value coincides with the level of light intensity at that point. Each element of the array corresponds to an element in the image and is called pixel. [4]

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad (1)$$

The notation of coordinates widely accepted by most of the books is shown in

equation (1) where the image has M rows and N columns determining the origin at the point *f (0,0).*

Figure 4 shows four version of the same picture where the difference is the number of pixels in each of them. This means that the pixel is only one division unit without a particular actual size. Only when the image resolution is given, a particular size to the pixel is assigned.

Figure 4. Digital Image of "Lenna" with different number of pixels

2.2 Classification of digital image

There are many kind of classification of digital image. A Basic classification should be: bitmap and vector images. [2]

Vector images are obtained based on lines, each responding to a mathematical equation. An image of this type is formed by controlled strokes coordinates. Vector graphics have the disadvantage that they do not have the level of detail of bitmaps. The advantage is that you can reduce and enlarge without losing quality since the lines are redrawn when resizing.

Bitmap images were described in point 2.1.2. These kinds of images are used in this project. Figure 5 shows differences between bitmap and vector images

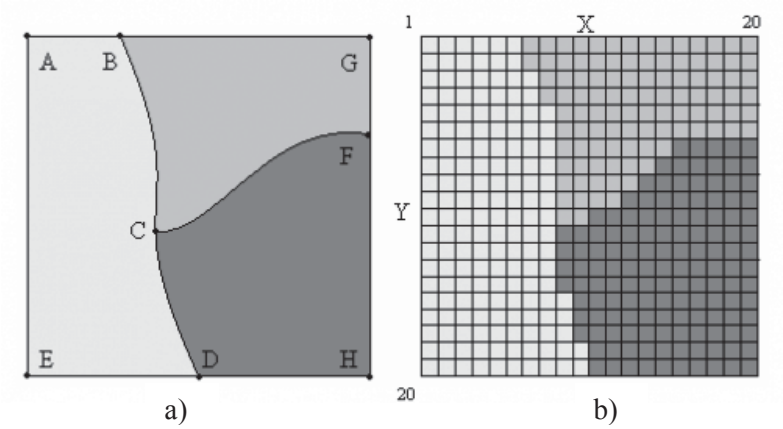

Figure 5. a) Vector and b) bitmap images

Within bitmap images can be classified according to:

- Size: 2D and 3D images [6]. See Figure 6a.
- Palette: binary images, grayscale or color. See Figure 6b,c,d.

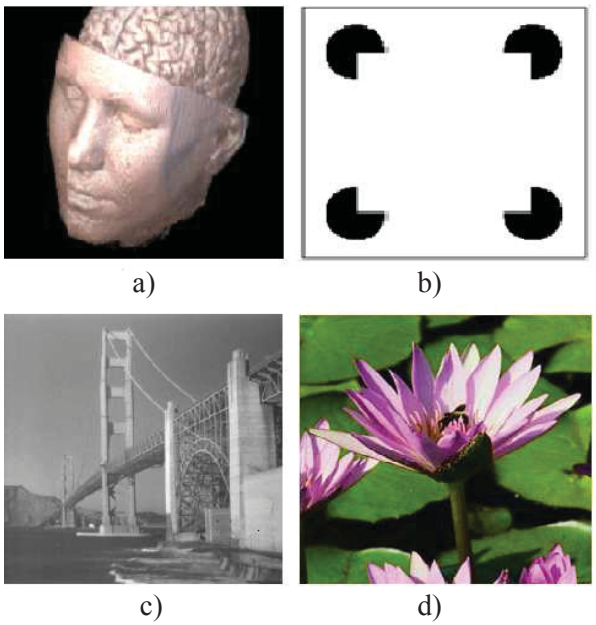

Figure 6. Different kinds of images: a) 3D b), binary, c) greyscale y d) and color

Greyscale and color images are used in this Project to evaluate the algorithms.

2.3 Color spaces

Color spaces are a defined range of colors that in combination with physical device, it allows representations of color in analog and digital way [7].

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers [4]

There are many types of model color but only the first 3 are important in this project. [8]

- Grayscale
- RGB
- HSV
- Others: YCbCr, HLS, CMY, etc.

2.3.1 Grayscale

An intensity scale is also known as monochrome or grayscale level and to a digital image is an MxN array of values where each pixel is a single sample containing the information of the image intensity.

In a grayscale image (figure 7a) each pixel has a brightness value between 0 (black) to 1 (white). Commonly, this mode uses up to 256 shades of gray (8 bits per sampled pixel). Another way of representation is as percentage (figure 7.b)

The 3 characteristics that can define a color are hue (color), value (lightness or darkening) and saturation (color purity). Thus the conversion of a color image to a grayscale image is not performed in a unique way, however in its most common approach [8], it is to retain information on the brightness and discard the values of hue and saturation. Assuming the colors red, green and blue are signs of light, the approximation of an image in grayscale from a color image is given by equation (2.3) where 0

is the value of less intensity, referring to the color black and 1 is the value of greater brightness or white.

$$GRAY = (0.30 \cdot R) + (0.59 \cdot G) + (0.11 \cdot B) \quad (2.3)$$

(a) (b)

Figure 7. a) Greyscale images y b) Grey levels of greyscale images

2.3.2 RGB model

An RGB image is defined as an array of 3xMxN pixels where each pixel corresponds to the red, green and blue components of a color image. The main purpose of the RGB model is the sensing, representation and display of images in electronic devices such as televisions, computers, cell phones, etc. [4]

The RGB model can be viewed as a stack of 3 scale image intensities to be displayed on a color monitor (which has 3 color inputs, red, green and blue). Colors red, green and blue are known as primary colors, and the combination of these different intensities in colors produces human visible spectrum. Figures 8 and 9 show a 3D representation of the RGB model.

Generally the intensity of each of the components is measured on a scale from 0 to 255 (1 byte per component)

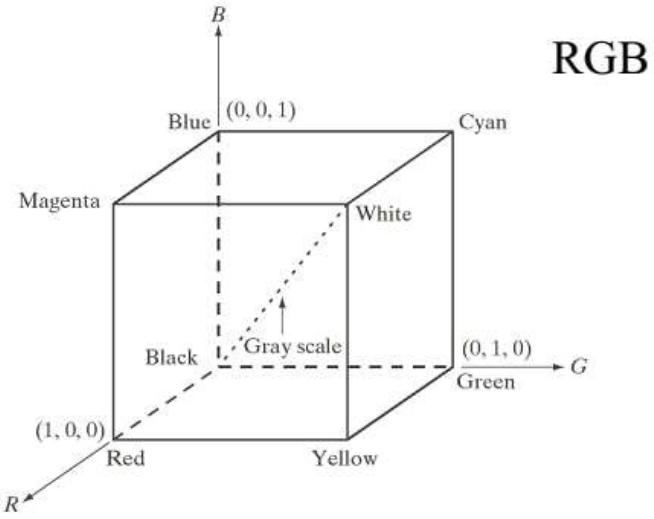

Figure 8. Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1,1,1)

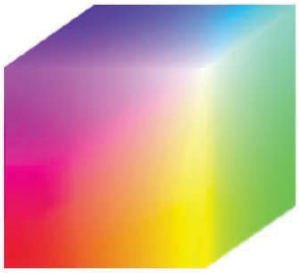

Figure 9. RGB 24-bits color cube

This model is the most used to display digital images on a screen in the current formats so it is very important in the image processing.

2.3.3 HSV model

The HSV model is based on the human perception of color and describes, according to CIE [quote]:

- Hue: The "attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them".

- Saturation: Colorfulness of an area judged in proportion to its brightness.

- Brightness: The "attribute of a visual sensation according to which an area appears to emit more or less light".

The HSV color model is based in the RGB model, but it is a cylindrical coordinate model. It uses the following three components [9]:

- **H** (hue) is usually represented in a circumference, so the degree says the color of that pixel, but it is also used in a percentage way for some applications.

- **S** (saturation), the representation of this component is the distance from the cylinder axe.

- **V** (value, also called B, brightness), this is the component used in the transform. Usually, it is represented from 0 to 1 (also in Matlab). If the value is 0, it means that the pixel is black, regardless of the other two components (for this reason, the HSV model can also be interpreted and represented like a cone) (figure 10).

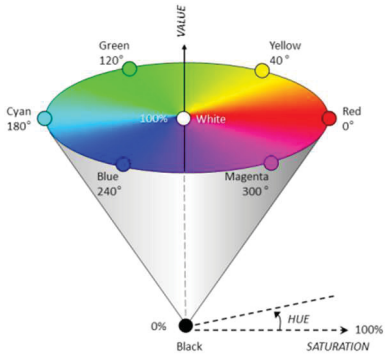

Figure 10. Cone of HSV model

The transformation from RGB to HSV is given by [9]:

$$H=\begin{cases} \text{Undefined} & \text{if MAX} = \text{MIN} \\ 60º\ x\ \frac{G-B}{MAX-MIN} + 0º, & \text{if MAX = R and G} \geq \text{B} \\ 60º\ x\ \frac{G-B}{MAX-MIN} + 360º, & \text{if MAX = R and G < B} \\ 60º\ x\ \frac{B-R}{MAX-MIN} + 120º, & \text{if MAX = G} \\ 60º\ x\ \frac{R-G}{MAX-MIN} + 120º, & \text{if MAX = B} \end{cases}$$

$$S=\begin{cases} 0, & \text{if MAX} = 0 \\ 1 - \frac{MIN}{MAX}, & \text{Otherwise} \end{cases}$$

$$V = MAX$$

Hence, the formula indicates that for a pixel, the Value or Brightness is the maximum value of any of the RGB components. For example, if R is 0.7, G is 0.5 and B is 0.1 (normalized), the value for this pixel is 0.7.

a) b)

Figure 11. a) Original picture. b) Component brightness of original picture

2.4 Digital Image processing

The digital image processing is the set of techniques applied to digital images in order to improve quality or facilitate the search for information [10].

The field that handles the processing of digital images is the digital image processing. Most processing techniques act treating the image as a two dimensions signal and then applying standard signal processing techniques of one dimension. Among the most common processing operations are:

- Intensity Transformation; exists for the spatial domain techniques that operate directly on the image pixels. The processes discussed in this report are denoted by the expression

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

where the function $f(x,y)$ is the input image, $g(x,y)$ is the output image (processed image) and T is an operator on f, which is an operator defined in a specific neighborhood (x,y) on a point (x,y).

Knowing that the main space to define neighborhoods about some point (x,y) approach is to use a square or rectangular region centered at (x,y) as shown below in the following scheme.

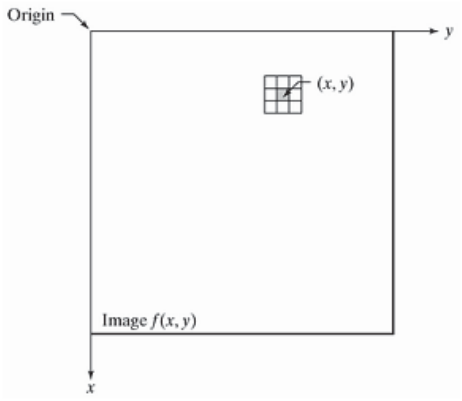

Figure 12. Spatial domain of digital image

The way to do this is to move starting pixel by pixel, that is, starting from the upper left corner as it moves, different neighborhoods are included. The operator T is applied to each location (x,y) thus g output can be obtained at that location. Only the pixels in the neighborhood

are used to calculate the value of $g(x,y)$.

Intensity transformation functions; the simplest form of the transformation T is when the neighborhood size is 1x1 (one pixel). In this case, the value of $g(x,y)$ depends only on the intensity at that point f, and T becomes a transformation function intensities or gray levels.

As depend only on the intensity values, and not explicitly on (x,y), these functions are usually written in simplified form as;

$$s=T(r)$$

Where r denotes the intensity of f and s the intensity of g, both on a corresponding point (x,y) of the image [4].

Some of these techniques are;

- Linear:
    - Negative; Invert the order of the intensity values.
      $T(r)=L-1-r$
    - Brightness; change of the average intensity of the image
      $T(r)=r \pm B$ *(B is real number)*
    - Contrast; change of the dynamic range of the image.
      $T(r)=r \cdot B$ *(B is real number)*

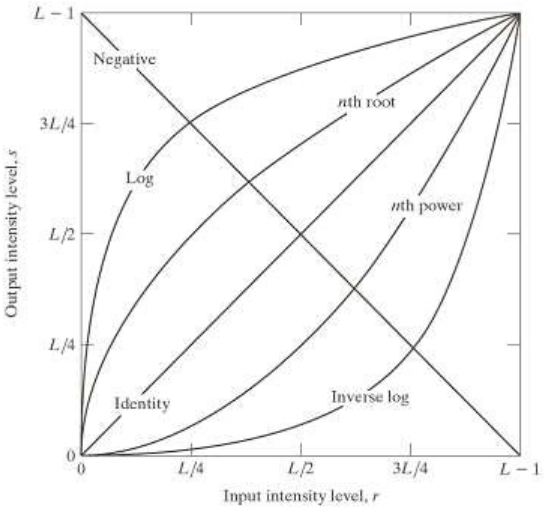

Figure 13. Basic transformations

- Nonlinear;
    - Log; It is used to display low levels of intensity with greater dynamic range.
      $T(r)=c\ log(1+r)$
    - Log power-law; It is similar to the log transformation. The advantage is the variety of transformations to modify the value of n
      $T(r)=c\ r^n$
    - Histogram equalization; it will be explained later
- Thresholding; change a grayscale image in binary image (black and white) through a threshold

$$T(r) = \begin{cases} 0 & if \quad r < T \\ 255 & if \quad r > T \end{cases}$$

➢ <u>Geometric transformation;</u> modify the spatial relationship between pixels. In terms of digital image processing a geometric transformation consists of two basic operations [11]:

1. A spatial transformation that defines the relocation of the pixels in the image plane.

2. Interpolation of gray levels, which are related to mapping the intensity values of the pixels in the transformed image.

Some of these techniques are symmetry, spin, rotation, etc.

- Transformations into frequency domain

  - Fourier Transformation
  - Filter

For this project, intensity transformation is used.

.

3 IMAGE ENHANCEMENT

Image enhancement processes consist of a collection of techniques that seek to improve the visual appearance of an image or to convert the image to a form better suited for analysis by a human or a machine [12].

In general, there is no general unifying theory of image enhancement at present because there is no general standard of image quality that can serve as a design criterion for an image enhancement processor

3.1 SMQT

The SMQT (Successive Mean Quantization Transform), is an algorithm that has the goal to get advantage of the whole dynamic range, but in a very different way as Histogram Equalization technique, which will be described later in 3.2. The SMQT aims to remove the disparity between sensors due to gain and bias [13]. The SMQT can be used to extend structure representation to an arbitrary predefined number of bits on arbitrary dimensional data.

The best results of the SMQT in an 8-bit image are obtained when using an 8 level SMQT.

The basic unit of the SMQT is the MQU (Mean Quantization unit), which consists in calculating the mean value of all the pixels in the image, then the mean is used to quantize the value of data into 0 or 1, depending if the value of the pixel is lower or higher than the mean. After doing this, the input is splitted in two.

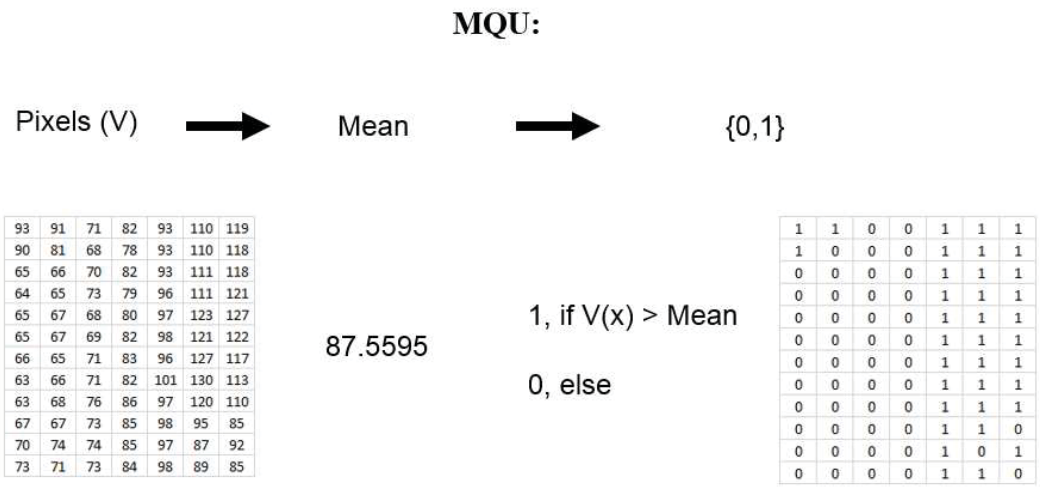

Figure 14. MQU operation

The SMQT can be seen as a tree of MQU operations, where a weight is given depending on the current level of the tree.

Weight = 2^{L-l}

Where L is the total number of levels and l is the current level.

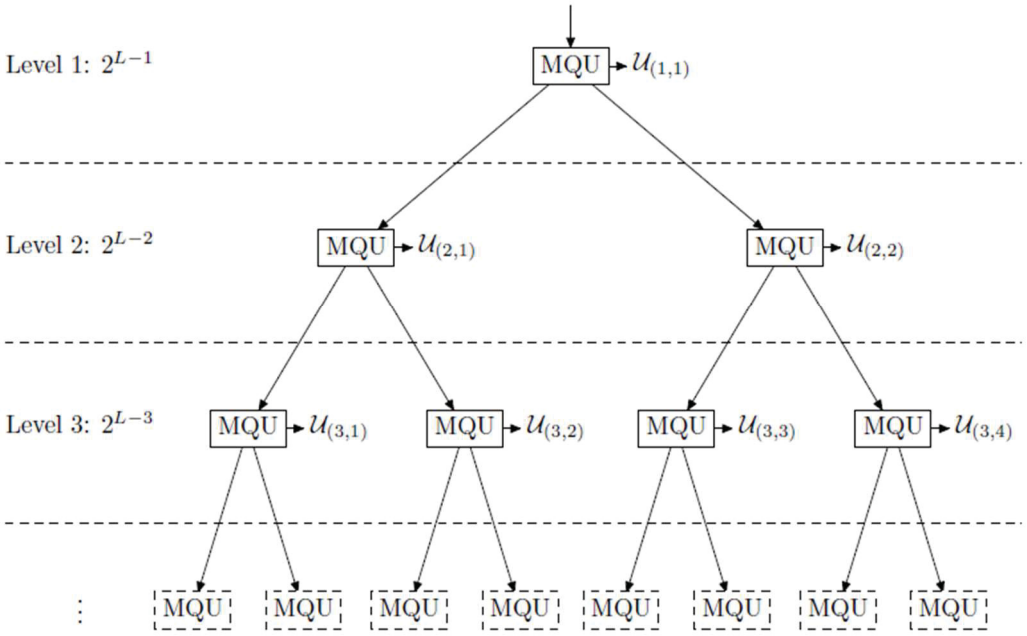

Figure 15. SMQT tree

In RGB images, the SMQT can be applied in two different ways. The first one, consists in apply the SMQT to the three RGB channels. Let $D_{RGB}(x)$ be all data values regardless of channel, then

$SMQT_L: D_{RGB}(x) \rightarrow M_{RGB}(x)$

This will result in a nonlinear contrast enhancement which preserves the order of the RGB values for each pixel, but with changed distances between the red, green and blue values within each pixel.

The second way is to apply it separately in each of the three channels. Let $D_R(x)$, $D_G(x)$ and $D_B(x)$ be all data values of the red, green and blue channel respectively, then

$SMQT_L: D_R(x) \rightarrow M_R(x)$

$SMQT_L: D_G(x) \rightarrow M_G(x)$

$SMQT_L: D_B(x) \rightarrow M_B(x)$

Finally, the enhanced pixels set $M_{RGB*}(x)$ is found by concatenating the channel sets $M_R(x)$, $M_G(x)$ and $M_B(x)$. This will result in a nonlinear contrast enhancement which neglects the order of the RGB values. For example, if an image is presented which is heavily biased toward one color, this technique reduces the influence of that color. Hence, this approach may work as a color corrector for some images. Nevertheless, in other cases, this color correction may generate color artifacts which will exaggerate color shifts or reduce color saturation. (10)

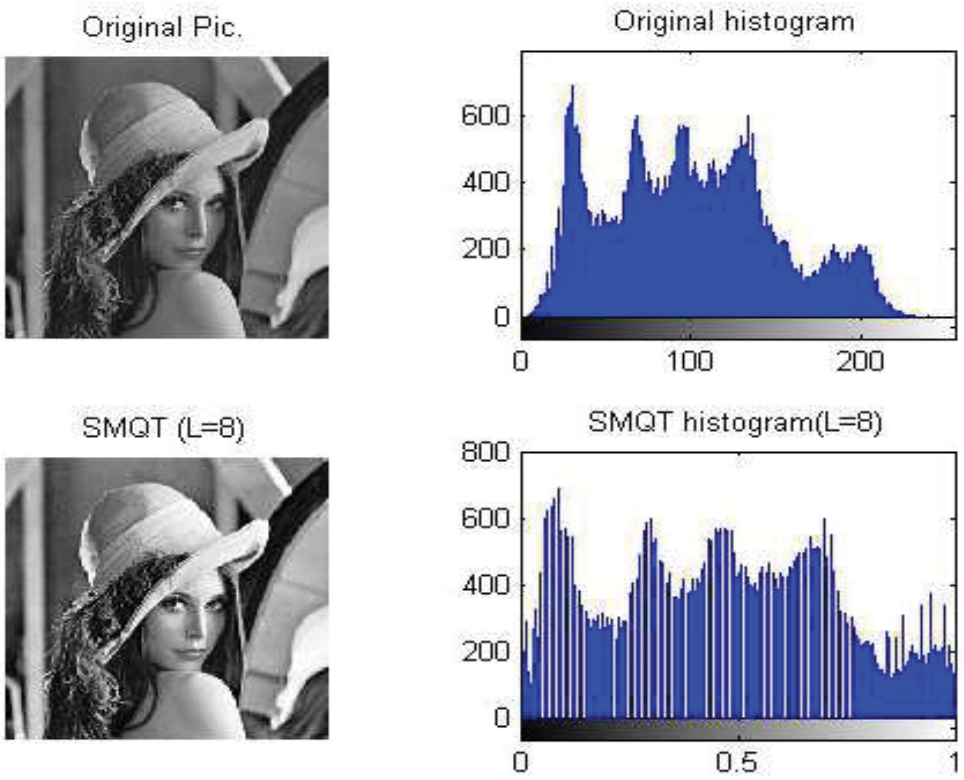

Figure 16. Grayscale image; Original and after SMQT

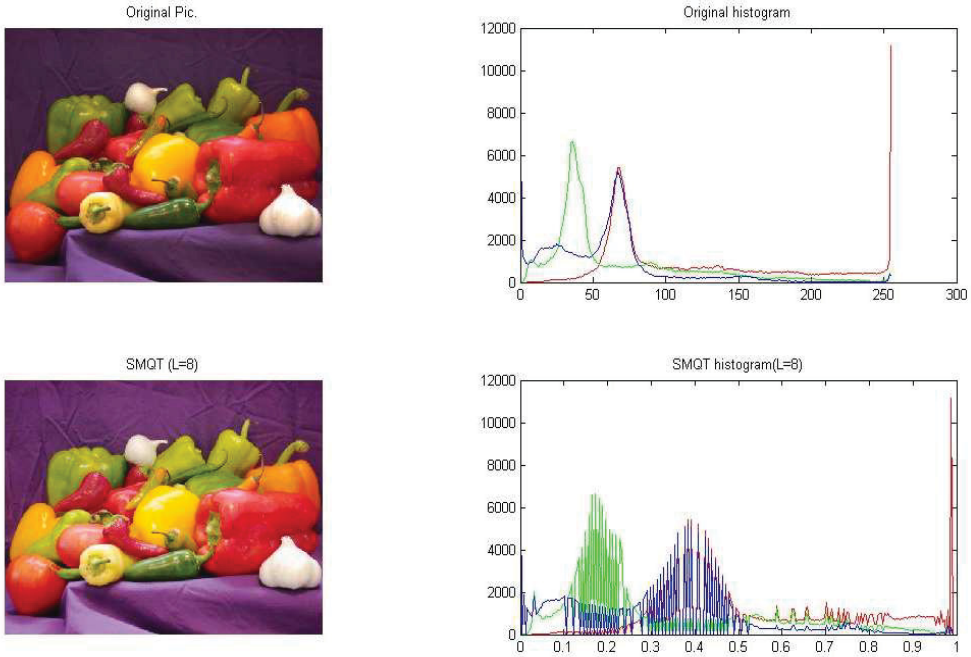

Figure 17. Color image. Original and after SMQT

3.2 Histogram Equalization

The histogram of an image is the representation of the number of pixels that have every value of color. In a greyscale image, it is usually represented as a graphic of the grey values, and in RGB images, the representation is done with three graphics, one for every color component (red, green and blue). [14]

To avoid dependence between the number of pixels or the number of quantization levels and the size of the histogram, usually the histogram axis are normalized between 0 and 1. This is the reason that the axes units did are show up.

One way to compare histograms is by Bhattacharya distance [15]. This distance is a factor of similarity between two vectors (in this case, histograms). Is between 0 and 1, with 0 being the histograms if nothing and 1 appear if they are the same.

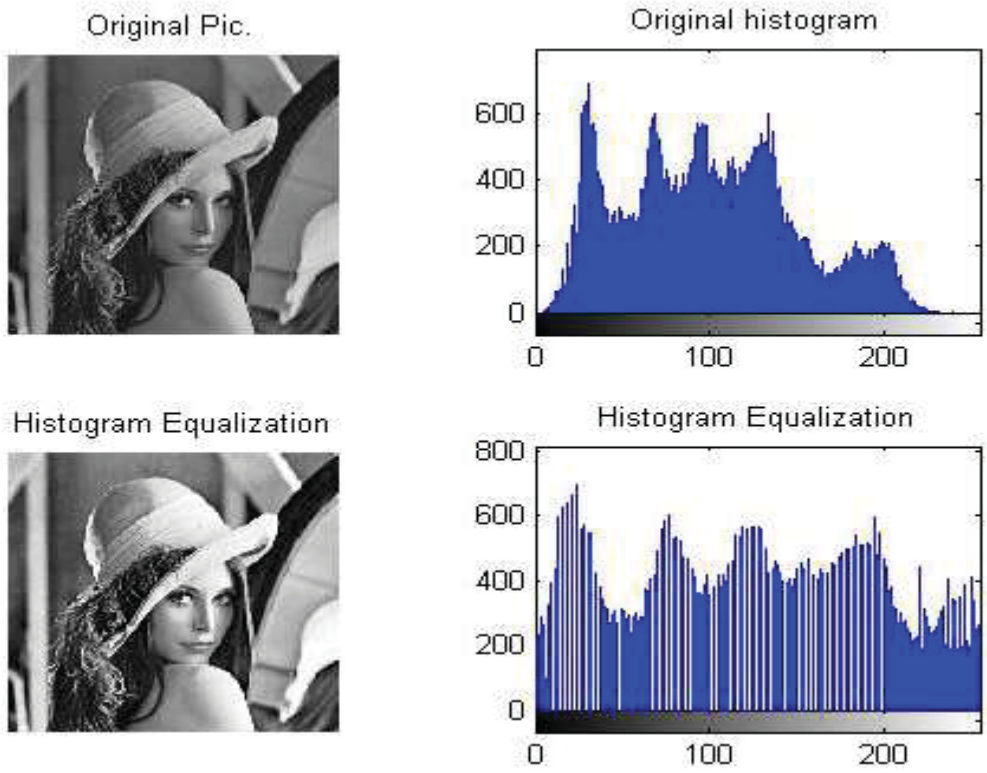

Figure 18. Grayscale image. Original and after Histogram Equalization

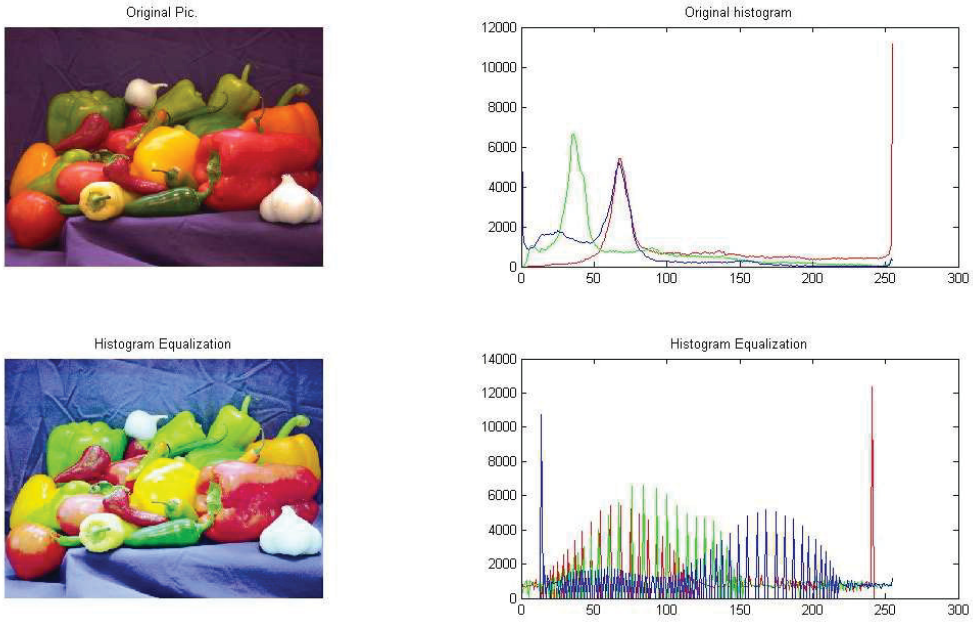

Figure 19. Color image. Original and after Histogram Equalization

Histogram equalization is a method created to get advantage of the whole dynamic range in any image. This technique consists in evaluating the probability of every level, and then reassigns a new level based on this probability. Ideally, the result would be a flat histogram, where all levels have the same probability, but practical results indicate that this is not true, because of the discrete nature of the data. The new image will have more contrast than the original picture.

-Implementation:

The histogram may be interpreted as a probability density function.

However, an image is a discrete function, then the cumulative distribution function is applied, so the accumulated histogram is obtained using the following approach.

$$f[m] = \frac{255}{N \times M} \cdot \sum_{k=0}^{m} h[k]$$

Once the original histogram is known, the values of the pixels are changed, based on the original probability, so now all values are spread over the histogram.

The histogram equalization is a good technique if the image has a wide variability in grey levels, but it should not be applied in images with bimodal nature. Often, histogram equalization produces unrealistic effect in image, but it can be very useful for some applications, as in medicine for x-ray images, satellites or thermal images.

Figure 20. Thermal image shows a fault with an industrial electrical fuse block.

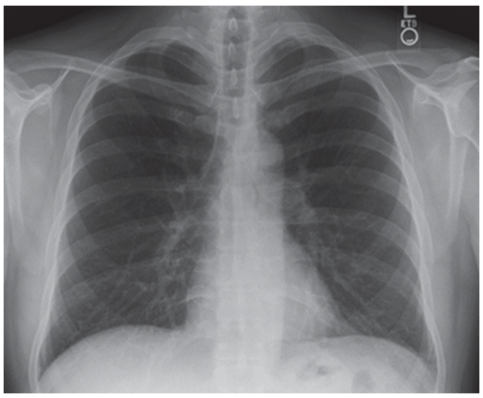

Figure 21. X-ray image

Using ‘histeq’ Matlab function, the shape obtained of the histogram is flatter, it is because the creators did not apply the mathematical definition, but the theoretical definition, so the goal of this function is to make the histogram as flat as possible (the source code of this function can be found in the annex).

Comparing the results, the differences in the new histogram are more or less evident depending on the image, but looking at the new image, it is hard for the human eye to appreciate great differences between the results of both functions.

Histogram equalization in RGB images is possible, but it has to be applied to each component, which can lead to have an image with artificial colors, or some weird results.

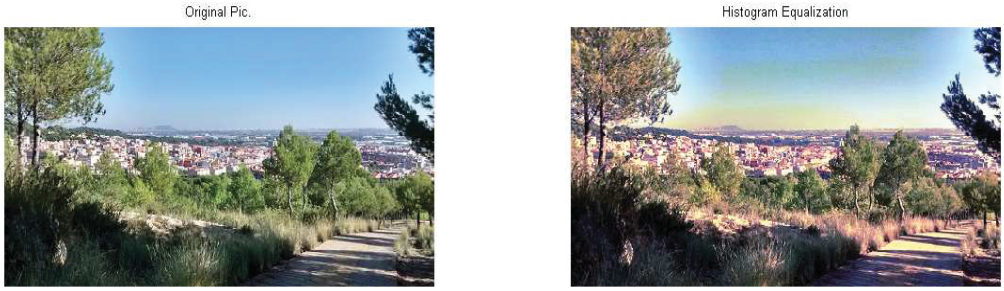

Figure 22.Example of histogram equalization applied to components RGB in a color image

3.3 V Transform

The V transform gets advantage from the HSV model. The V transforms contains information about the brightness, so it can be modified in color images without changing the color. It is an advantage because it means that it changes color images operating only in one component, so the computational cost is lower than operating in the three channels. A part from this, this transform requires less computational power than SMQT for its simplicity.

The first step is converting the RGB image into an HSV image.

Second step consists in extracting the V component and making a sorted vector with the V values.

Then, the sorted vector is divided in N segments of the same length. For each interval a start and a stop value is defined.

Finally, a linear transformation is performed in every segment, in order to spread the brightness from the start to the stop values.

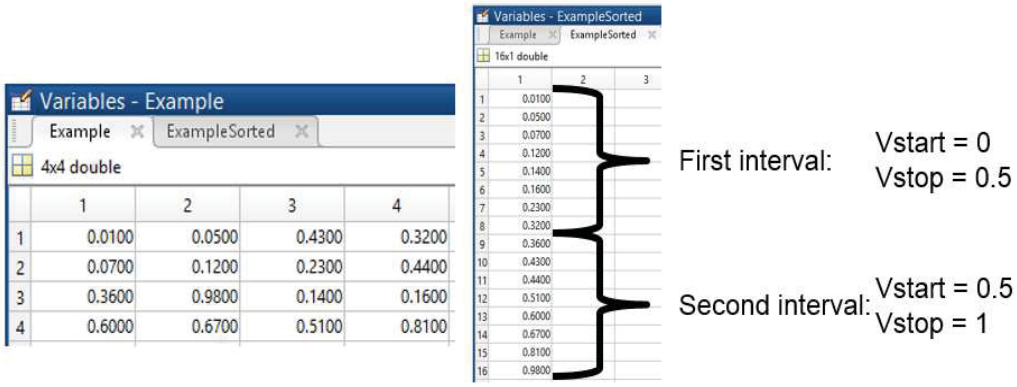

Figure 23. Schematic of the V transform

V transform can also be applied to greyscale image, if the image is considered as the V component itself.

Results:

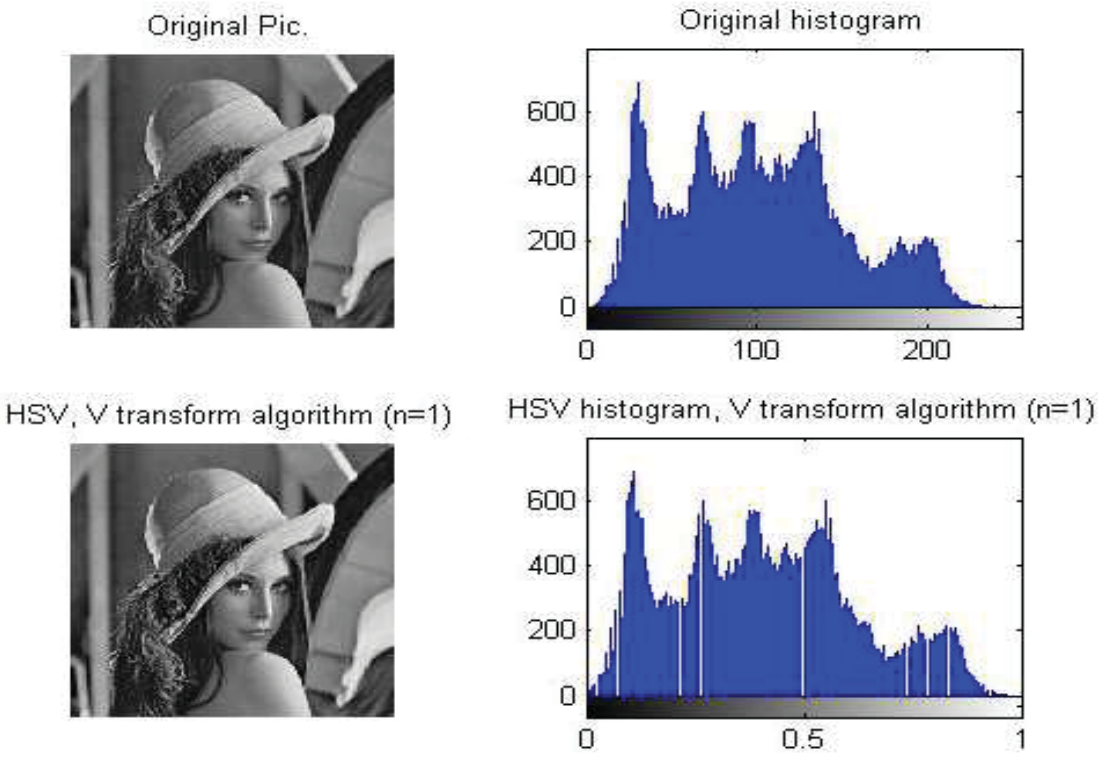

Figure 24. Grayscale image. Original and after V transform algorithms with n=1

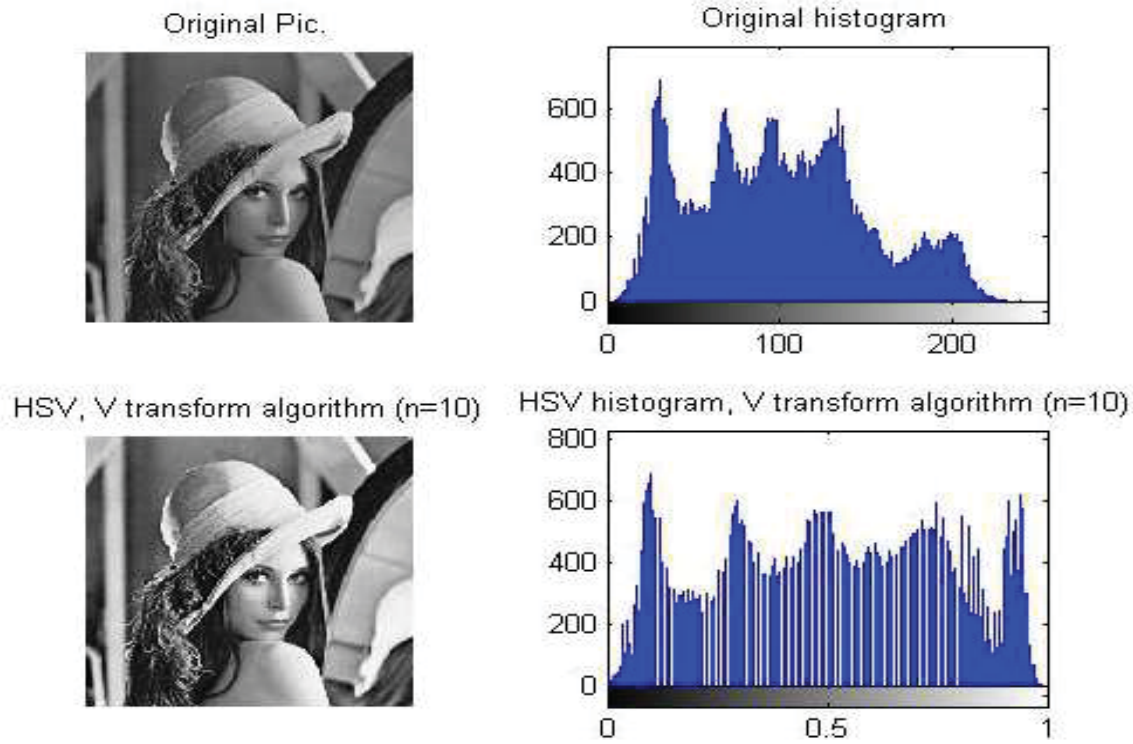

Figure 25. Grayscale image. Original and after V transform algorithms with n=10

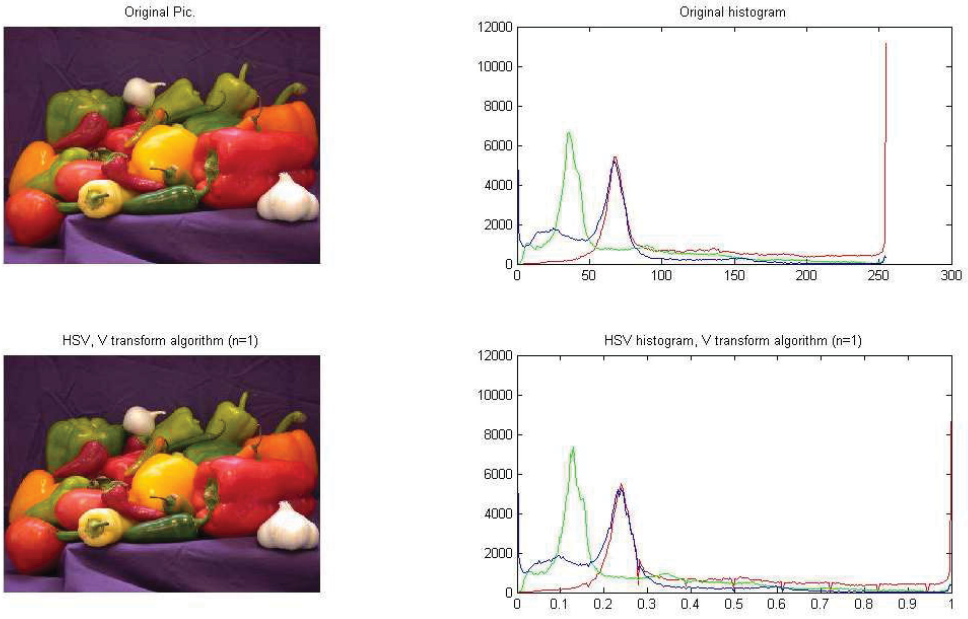

Figure 26. Color image. Original and after V transform algorithms n=1

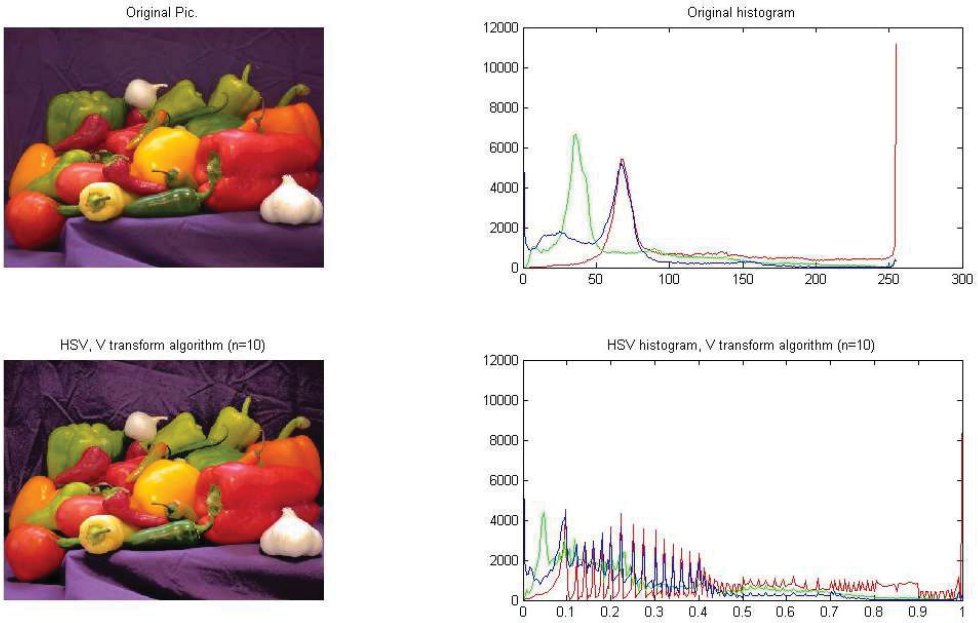

Figure 27. Color image. Original and after V transform algorithms n=10

This algorithm is good for n=1, but it does not introduce big changes if the original image has much colors. For higher values of n, the transformed image has more contrast and gets strange effects, because the algorithm does not consider the initial composition of the picture, it spreads all the values indistinctly (figure 28). This can be good for some applications, as it happened in histogram equalization, but in this case, the color is not changed, only its brightness.

a)

b)

c)

d)

d)

Figure 28. Colour image. Comparative of V transform algorithms between; a) original. b) n=1. c) n=2. d) n=5. e) n=50

4 DESIGN

4.1 MATLAB

MATLAB (short for MATrix LABoratory) is a mathematical software that offers integrated development environment (IDE) with a programming language itself (language M). It is available for Unix, Windows and MacOS X platforms.

Among its high potential, basic functions worthy of mention are matrix manipulation, data and functions representation, implementation of algorithms, creation of user interfaces (GUI) and communication with other programming languages and other hardware devices.

The software package MATLAB has two additional tools that expand their benefits: Simulink (multidomain simulation platform) and GUIDE (editor user interfaces - GUI). Additionally, you can extend the capabilities of MATLAB with toolboxes. Toolboxes currently cover most major areas of the world of engineering and simulation, such as digital processing toolbox Images (IPT hereinafter), the signal processing toolbox, the communications toolbox, etc.

MATLAB provides an interactive work environment whose basic working element are the matrices, allowing the numerical solution of problems in a much shorter time than in traditional programming languages such as Fortran, Basic or C languages, with the advantage that its own programming language is similar to traditional languages.

When working with matrices numerous variables can be described mathematically in a highly flexible and efficient way. For example, an image can be written as a matrix of pixels, a sound like a matrix jitter, and generally can be described with any linear matrix relationship between the components of a mathematical model.

4.2 SMQT

```
function A = alg_a(RGBimage, n)
%This algorithm transforms an RGB image to an HSV image, and
transforms the V component.
%
[row, column, d] = size(RGBimage);%We get the size of the image

if (d==3) %if the image has 3 dimensions, it's an rgb image
    HSVimage = rgb2hsv(RGBimage); %Transform from RGB to HSV
    V = HSVimage(:,:,3); %Get the component V (brightness)
else %if the image has 1 dimension, it is a gray-scale image
    V =double(RGBimage)/255;%In this case, we don't need to
transform
end

V=V(:); %The matrix of brightness is now a vertical vector

[Vsorted, ix] = sort(V); %Sorting the vector

s = (row*column)/n; %size of the intervals

i=0; %initializing i

h=[]; %initialaizing h

% now, there is a loop to process every interval
while (i < n)
    i=i+1;

    z = Vsorted(((floor(s*(i-1))+1)):floor(s*i)); %we define the
interval

    Vstart = (s*(i-1))/(row*column); %We define the start and the
end of the interval
    Vstop = (s*i)/(row*column);

    %linear transform for each segment
    r=z-z(1);

    f = (1/n)/(r(size(r,1)));

    g = r*f;

    if(isnan(g(1)))
        g = r + Vstop;
    else
        g = g + Vstart;
    end

    h=vertcat(h,g); %Bulding the transformed vector

end

m(ix)=h; %We "reverse the sorting" of the vector, with the
transformed values

m=m(:);
```

```
if(d==3) %resizing the vector into a matrix and assigning the new V
component
    HSVimage(:,:,3) = reshape(m,row,column);
    A=hsv2rgb(HSVimage);
else
    A=reshape(m,row,column);
end

return;

end
```

4.3 Histogram Equalization Matlab

```
%% alg_hm.m
% Function to call Histogram Equalization Matlab.
% Input:
%   - OriginalImage is original image
% Output:
%   - HistogramEqualization is enhancement image
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%
function HistogramEqualization = alg_hm(OriginalImage)

[row, column, d] = size(OriginalImage);

i=0;

while(i<d)
    i=i+1;
    HistogramEqualization(:,:,i) = histeq(OriginalImage(:,:,i));
end

end
```

4.4 Histogram Equalization own function

```
%% alg_h.m
% Function to calculate Histogram Equalization
% Input:
%   - Input is original image
% Output:
%   - Output is image enhancement
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%
function Output = alg_h(Input)

% To know number of row and column (total number of pixels)
[row, column, d] = size(Input);
```

```
i=0;

while(i<d)%note that the loop is for using with RGB images
    i=i+1;
    % Calculate of the histogram
    histo=imhist(Input(:,:,i));

    % Convert Input in double
    Input(:,:,i) = double (Input(:,:,i));

    % Calculate probability of grey level pixel
    probability=histo./(row*column);

    % Accumulated probability and weights
    equalizer = cumsum(probability)*256;

    %Replace grey level pixels in function of equalizer
    Output(:,:,i)=equalizer(Input(:,:,i)+1);
end

%Convert to 8 bits unsigned intiger data
Input = uint8(Input);
Output = uint8(Output);

end
```

4.5 V Transform

```
%% alg_a.m
% Function to transforms an RGB image to an HSV image, and
transforms
% the V component.
% Input:
%   - RGBimage is original image
%   - n is number of intervals
% Output:
%   - A is image enhancement
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%
function A = alg_a(RGBimage, n)

[row, column, d] = size(RGBimage);%We get the size of the image

if (d==3) %if the image has 3 dimensions, it's an rgb image
    HSVimage = rgb2hsv(RGBimage); %Transform from RGB to HSV
    V = HSVimage(:,:,3); %Get the component V (brightness)
else %if the image has 1 dimension, it is a gray-scale image
    V =double(RGBimage)/255;%In this case, we don't need to
transform
end

V=V(:); %The matrix of brightness is now a vertical vector
```

```
[Vsorted, ix] = sort(V); %Sorting the vector

s = (row*column)/n; %size of the intervals

i=0; %initializing i

h=[]; %initialaizing h

% now, there is a loop to process every interval
while (i < n)
    i=i+1;

    int = Vsorted(((floor(s*(i-1))+1)):floor(s*i)); %we define the
interval

    Vstart = (s*(i-1))/(row*column); %We define the start and the
end of the interval
    Vstop = (s*i)/(row*column);

    %linear transform for each segment
    r=int-int(1);

    f = (1/n)/(r(size(r,1)));

    g = r*f;

    if(isnan(g(1)))
        g = r + Vstop;
    else
        g = g + Vstart;
    end

    h=vertcat(h,g); %Bulding the transformed vector
end

m(ix)=h; %We "reverse the sorting" of the vector, with the
transformed values

m=m(:);

if(d==3) %resizing the vector into a matrix and assigning the new V
component
    HSVimage(:,:,3) = reshape(m,row,column);
    A=hsv2rgb(HSVimage);
else
    A=reshape(m,row,column);
end
return;
end
```

4.6 Auxiliary functions

4.6.1 Main

```
%% ImageEnhancement.m
% Function applies SMQT, Histog Equalization(own function and Matlab
% function), and HSV transform.
% To use:
%   nameImagen = imread('c:\matlab\nameImagen.tif');
% Input:
%   - OriginalImage is the original image
% Output:
%   Print Original and histogram picture and Images enhancement and
%   histogram enhacement
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%
function ImageEnhancement(OriginalImage)

OriginalImage = uint8(OriginalImage);

%Call SMQT function
img_m=alg_m(OriginalImage,1,8);%variables: Image, l, L. l must be
one. L is the number of levels of the SMQT

%call Histogram equalization (own function)
img_h=alg_h(OriginalImage);

%Call Histogram equalization (matlab)
img_hm=alg_hm(OriginalImage);

%Call HSV, V transform algorithm, n=1
img_a1=alg_a(OriginalImage,1);

%Call HSV, V transform algorithm, n=10
img_a10=alg_a(OriginalImage,10);

%% PRINT ON SCREEN
% Print on screen a general comparative of all images
generalComp(OriginalImage,img_m,img_h,img_hm,img_a1,img_a10);

% Print on screen an individual comparative between original image
and the
% others
individualComp(OriginalImage,img_m,img_h,img_hm,img_a1,img_a10)

end
```

4.6.2 Split RGB components

```
%% splitRGB.m
```

```
% Function to split all components of RGB image and then return
count and
% bin of histogram of each component
% Input:
%   - image is de original image
% Output:
%   - yRed, yGreen, yBlue are histogram counts of components
%   - xr, xg, xb are bin locations of components
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%


function [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(image)

%Split into RGB Channels
Red = image(:,:,1);
Green = image(:,:,2);
Blue = image(:,:,3);

%Get histValues for each channel
[yRed, xr] = imhist(Red);
[yGreen, xg] = imhist(Green);
[yBlue, xb] = imhist(Blue);

end
```

4.6.3 Print on screen

Individual comparative between original image and transform image

```
%% individualComp.m
% Function to print on screen an individual comparative between
original
% image and the others.
% Input:
%   - OriginalImage,img_m,img_h,img_hm,img_a1,img_a10 are image
% Output:
%   Print Original and histogram picture and Images enhancement and
%   histogram enhacement
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014

function
individualComp(OriginalImage,img_m,img_h,img_hm,img_a1,img_a10)
%Original vs SQMT
figure;
subplot(2,2,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,2,3);imshow(img_m);title('SMQT (L=8)');
if (size(OriginalImage,3)<=1)% if image is in greyscale
    subplot(2,2,2);imhist(OriginalImage);title('Original
histogram');
    subplot(2,2,4);imhist(img_m);title('SMQT histogram(L=8)');
else % if image is in color
```

```
[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,2,2);
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_m); subplot(2,2,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('SMQT histogram(L=8)');
end

% Original vs Histogram Equalization
figure;
subplot(2,2,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,2,3);imshow(img_h);title('Histogram Equalization');
if (size(OriginalImage,3)<=1)% if image is in greyscale
    subplot(2,2,2);imhist(OriginalImage);title('Original
histogram');
    subplot(2,2,4);imhist(img_h);title('Histogram Equalization');
else % if image is in color

[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,2,2);
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_h); subplot(2,2,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('Histogram Equalization');
end

% Original vs Histogram Equalization (Matlab)
figure;
subplot(2,2,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,2,3);imshow(img_hm);title('Histogram Equalization
(Matlab)');
if (size(OriginalImage,3)<=1)% if image is in greyscale
    subplot(2,2,2);imhist(OriginalImage);title('Original
histogram');
    subplot(2,2,4);imhist(img_hm);title('Histogram Equalization
(Matlab)');
else % if image is in color

[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,2,2);
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_hm); subplot(2,2,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('Histogram Equalization (Matlab)');
end

% Original vs HSV (n=1)
figure;
subplot(2,2,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,2,3);imshow(img_a1);title('HSV, V transform algorithm
(n=1)');
if (size(OriginalImage,3)<=1)% if image is in greyscale
    subplot(2,2,2);imhist(OriginalImage);title('Original
histogram');
    subplot(2,2,4);imhist(img_a1);title('HSV histogram, V transform
algorithm (n=1)');
else % if image is in color

[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,2,2);
```

```
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_a1); subplot(2,2,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('HSV histogram, V transform algorithm (n=1)');
end

% Original vs HSV (n=10)
figure;
subplot(2,2,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,2,3);imshow(img_a10);title('HSV, V transform algorithm
(n=10)');
if (size(OriginalImage,3)<=1)% if image is in greyscale
    subplot(2,2,2);imhist(OriginalImage);title('Original
histogram');
    subplot(2,2,4);imhist(img_a10);title('HSV histogram, V transform
algorithm (n=10)');
else % if image is in color

[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,2,2);
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_a10); subplot(2,2,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('HSV histogram, V transform algorithm (n=10)');
end

end
```

General comparative between original image and all transform images.

```
%% generalComp.m
% Function to print on screen a general comparative of all images
% Input:
%   - OriginalImage,img_m,img_h,img_hm,img_a1,img_a10 are image
% Output:
%   Print Original and histogram picture and Images enhancement and
%   histogram enhacement
% Julián Calderón González
% Òscar Daniel Carmona Salazar
% 10/12/2014
%%

function
generalComp(OriginalImage,img_m,img_h,img_hm,img_a1,img_a10)
% Images
figure;
subplot(2,3,1);imshow(OriginalImage);title('Original Pic.');
subplot(2,3,2);imshow(img_m);title('SMQT (L=8)');
subplot(2,3,3);imshow(img_h);title('Histogram Equalization');
subplot(2,3,4);imshow(img_hm);title('Histogram Equalization
(Matlab)');
subplot(2,3,5);imshow(img_a1);title('HSV, V transform algorithm
(n=1)');
subplot(2,3,6);imshow(img_a10);title('HSV, V transform algorithm
(n=10)');
```

```
% Histograms
figure;
% if image is in greyscale
if (size(OriginalImage,3)<=1)
    subplot(2,3,1);imhist(OriginalImage);title('Original
histogram');
    subplot(2,3,2);imhist(img_m);title('SMQT histogram(L=8)');
    subplot(2,3,3);imhist(img_h);title('Histogram Equalization');
    subplot(2,3,4);imhist(img_hm);title('Histogram Equalization
(Matlab)');
    subplot(2,3,5);imhist(img_a1);title('HSV histogram, V transform
algorithm (n=1)');
    subplot(2,3,6);imhist(img_a10);title('HSV histogram, V transform
algorithm (n=10)');
% if image is in color
else

[yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(OriginalImage);subplot(2,3,1);
plot(xr, yRed, 'r', xg, yGreen, 'g', xb, yBlue, 'b');title('Original
histogram');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_m); subplot(2,3,2);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('SMQT histogram(L=8)');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_h); subplot(2,3,3);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('Histogram Equalization');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_hm); subplot(2,3,4);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('Histogram Equalization (Matlab)');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_a1); subplot(2,3,5);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('HSV histogram, V transform algorithm (n=1)');
    [yRed,xr,yGreen,xg,yBlue,xb]=splitRGB(img_a10); subplot(2,3,6);
plot(xr, yRed, 'Red', xg, yGreen, 'Green', xb, yBlue,
'Blue');title('HSV histogram, V transform algorithm (n=10)');
end
end
```

5 RESULTS

Practical results of the techniques explained previously applied to popular images in image processing can be found in this chapter, divided by greyscale and color images. The histograms can also be found here. All the images are obtained with MATLAB.

Greyscale images:

Figure 29. Results of Lenna greyscale image

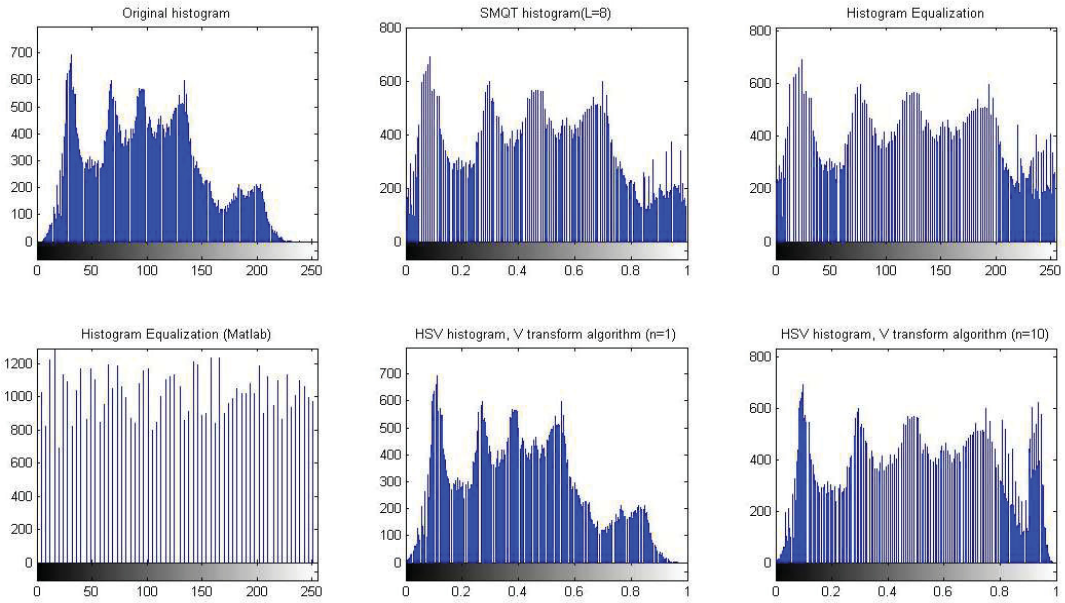

Figure 30. Histogram results of Lenna greyscale image

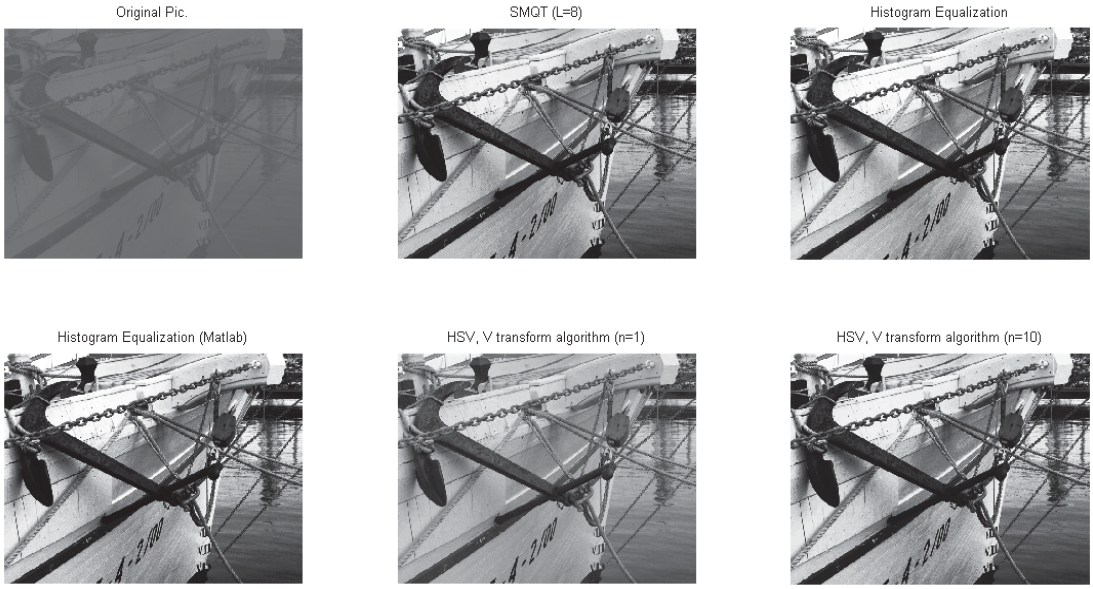

Figure 31. Results of Boat Grayscale image

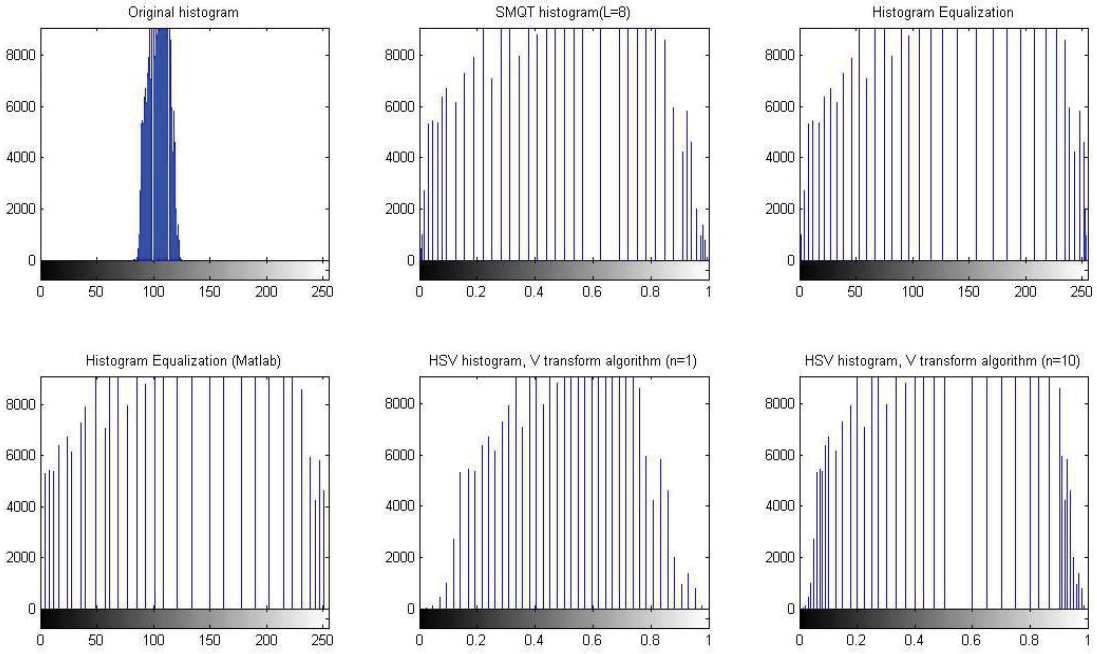

Figure 32. Histogram results of Boat Grayscale image

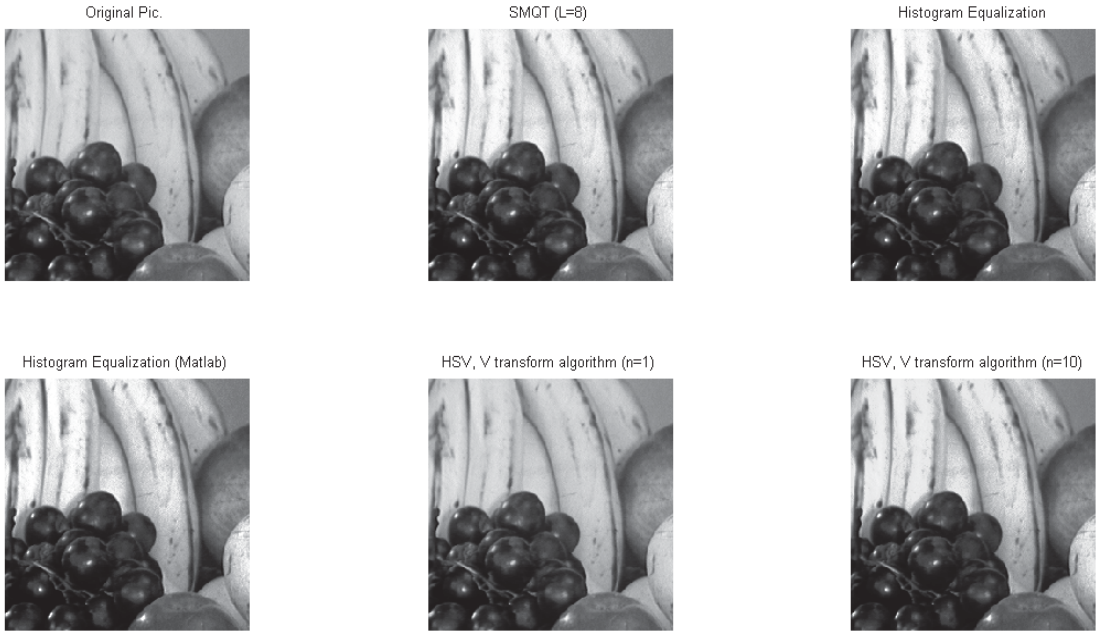

Figure 33. Results of Fruits Grayscale image

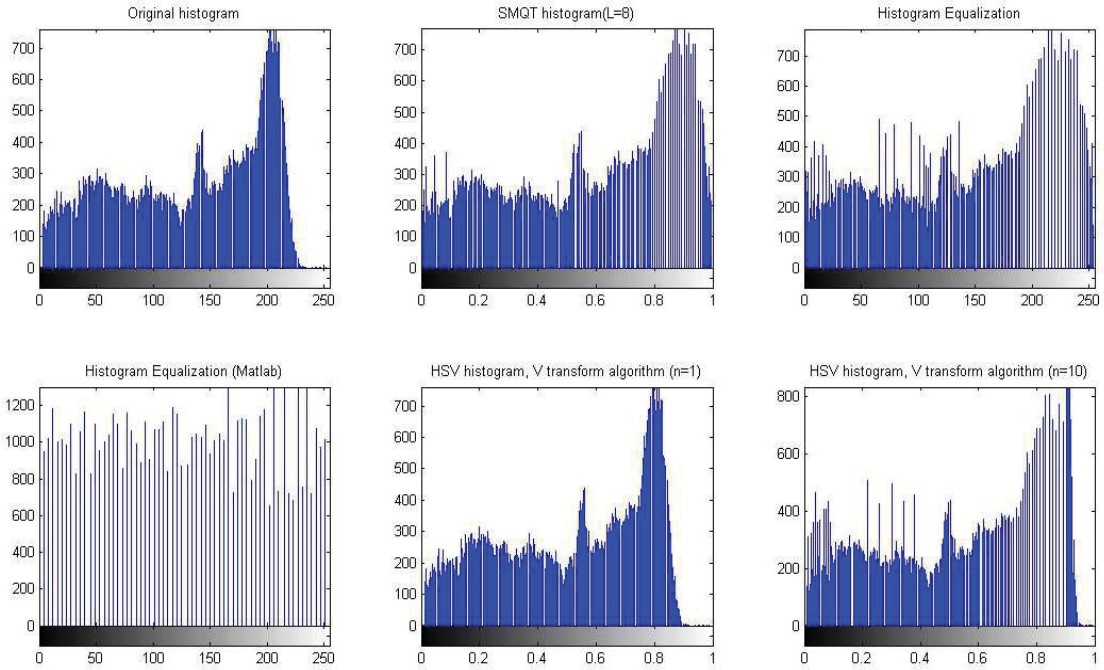

Figure 34. Histogram results of fruits Grayscale image

Color images:

Figure 35. Colour image

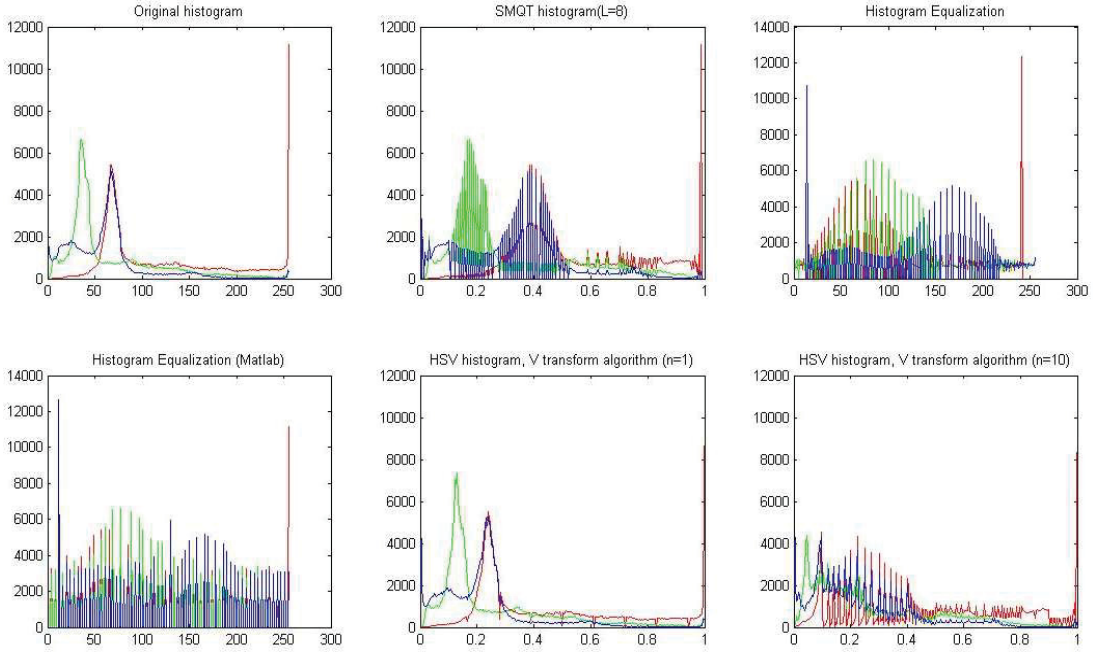

Figure 36. Histogram results of color image

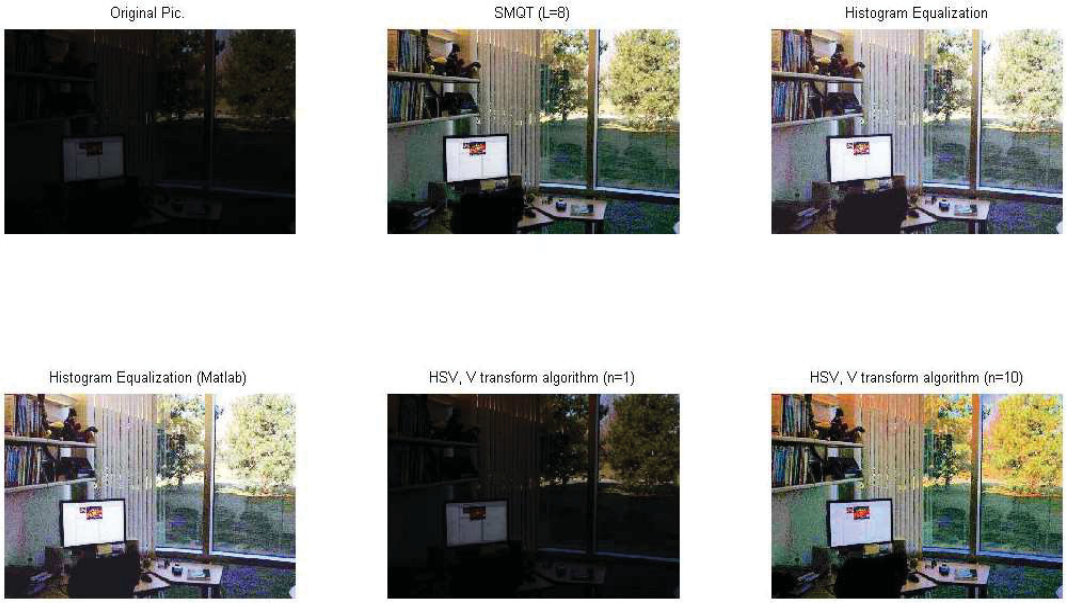

Figure 37. Colour image 2

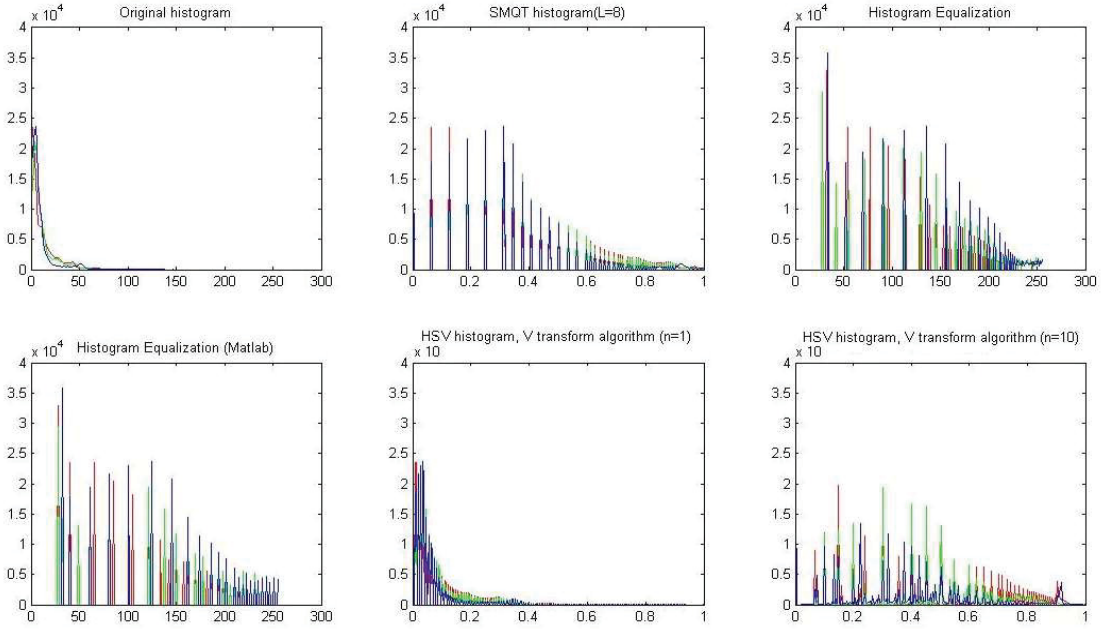

Figure 38. Histogram results of color image 2

6 CONCLUSIONS

On one hand, it looks clear that Histogram equalization is not a good way to enhance color images, the results show strange colors. In the other hand, Histogram Equalization is a very good technique to increase the contrast and enhance grayscale images, it is simple, requires low computational power and obtains good results.

In general, the best looking results for color images are the SMQT and HSV transform (with n=1). The contrast is increased, and the original colors continue in the image. In comparison, the SMQT method is good for images having a good light, but the transformation of the V component in HSV algorithm gets very interesting results in images with low light.

The SMQT obtains the best results, so it would be the appropriated technique to use in systems that require high accuracy and have good performance.

The main advantage of this algorithm regarding SMQT is that it requires a low computational power. The disadvantage is that results are not as good as in SMQT.

The elaboration of this project has been a great experience to end our career, and has been very useful to get more experience with MATLAB and a good introduction to image processing.

7 REFERENCES

1. FITER, E. L. **Descripción, comparación y ejemplos de uso de las funciones de la Toolbox de procesasdo digital de imagenes en Matlab**. Universidad Politécnica de Madrid. Madrid. 2012.
2. PEREZ., L. G. Tecnologías de la Información:Edición de imagenes. Available in: <http://platea.pntic.mec.es/~lgonzale/tic/imagen/imagen.html>. Access: 2014.
3. KHOSROW-POUR, M. **The Encyclopedia of Information Science and Technology**. third. ed. [S.l.]: [s.n.], 2014.
4. R.C. GONZÁLEZ, R. E. W. **Digital Image Processing.** Pearson Prentice Hall.: Pearson Prentice Hall, 2008. Texto básico y completo, con muchos ejemplos., 2008.
5. LASSO, S. About web site. Available in: <http://arte.about.com/od/Que-es-el-arte/ss/Colores-primarios-secundarios-y-terciarios.htm>. Access: December 2014.
6. CORP, A. S. 3D-doctor. FDA 510K Cleared, vector-based 3d imaging, modeling and measurement software. Available in: <http://www.3d-doctor.com/volume.html>.
7. WIKIPEDIA. Wikipedia. **Color space**, 2015. Available in: <http://en.wikipedia.org/wiki/Color_space>.
8. INCORPORATED., A. S. **PostScript Language Reference Manual. Chapter 6:** Rendering, 2nd ed.. [S.l.]: Addison-Wesley Publishing Company, 1990.
9. WIKIPEDIA. Wikipedia - Modelo_de_color_HSV, 2015. ISSN http://es.wikipedia.org/wiki/Modelo_de_color_HSV.
10. KHOSROW-POUR, M. **Encyclopedia of Information Science and Technology**. [S.l.]: Information Science Reference; 2 edition , 2008.
11. GRENIER, T. INSA Lyon. **Biomedical Imaging Research Lab**, 2015. Available in: <http://www.creatis.insa-lyon.fr/~grenier/wp-content/uploads/teaching/DIP/DIP-3DiscretProcessing.pdf>. Access: December 2014.
12. PRATT., W. K. **Digital image processing**. [S.l.]: Wiley-Interscience, 2001.
13. M. NILSSON, M. D. A. I. C. **The successive mean quantization transform**. Proc. IEEE Int. Conf. Acoust Speech, Signal Process. vol. 4. Mar. 2005, pp. 429–432.
14. VÉLEZ, J. . S. A. . M. A. B. . E. J. L. **Visión por computador**. [S.l.]: Dykinson S. L. , 2003.
15. SRIHARI, S. . T. C. . Z. B. . A. L. S. **Individuality of Numerals. In Proc. Of the Seventh International Conference of Document Análisis and Recognition (ICDAR 2003).** [S.l.]. 2003.
16. THE MATHWORKS, I. Matlab. Available in: <www.mathworks.com>.
17. R.C. GONZÁLEZ, R. E. W. S. L. E. **Digital image processing using MATLAB**. [S.l.]: Prentice Hall, 2004.
18. CHRIS SOLOMON, T. B. **Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab**. [S.l.]: Cranfield University, 2011.
19. EVA PART-ENADER, A. S. B. M. P. I. **The Matlab handbook**. England: Addison-Wesley, 1996.
20. CASTLEMAN, K. R. **Digital Imagin Processing**. [S.l.]: Printice-Hall, 1996.