Greyscale image enhancement using automatic contrast strecthing

Guillaume Lemaitre
ID student : 09295005
Heriot-Watt University, Universitat De Gerona, Universite De Bourgogne

October 16, 2009

Image Pocessing (B31SG)
Lecturer : Andrew Wallace

Abstract

Images provide by cameras can be of different qualities. This quality depends on specifications and prices of each camera. A weakness of cheap cameras is contrast. In this paper we introduce automatic contrast stretching based on transformation histogram using piecewise of logistic function. Our method improve contrast without need interaction with user.

1 Introduction

Contrast stretching is a basic tool to obtain image enhancement. It is used in many fields like digital photography or medical imaging to allow better distinction of features on images. The aim is to compute an algorithm allowing to enhance image automatically. This paper is organized as follow: Section 2 presents image enhancement using automatic contrast stretching and especially transformation, analysis and mapping histogram. In section 3, results and discussion are presented based on different images. A conclusion is given in the last section.

2 Our Method

2.1 Histogram Transformation

Firstly, we will define some notations. We call $f(x, y)$ the input image and $g(x, y)$ the output image. We call r and s, repsectively, the intensity of f and g at any point (x, y). Hence, an intensity transformation can be written:

$$s = T(r)$$

Secondly, we base our intensity transformation on a piecewise of exponential. Thus, we use logistic function:

$$T(x) = \frac{1}{1+e^{-\theta \times (x-m)}} \tag{1}$$

where θ is the slope and m is the point where the logistic function is centered. We will analyse the behaviour by changing parameters m and θ independently.
Hence, if m is variable and θ is constant, we obtain:

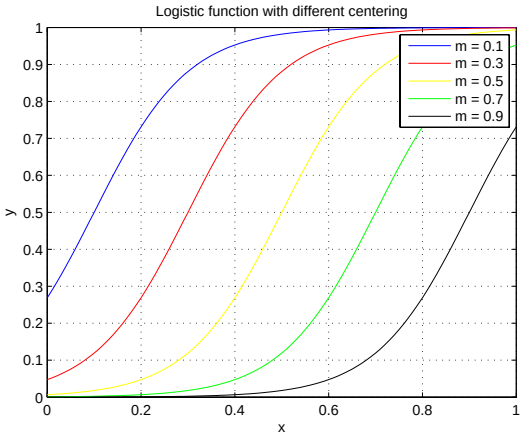

Figure 1: Logistic function for different centering

We can see that when we change parameter m, the center of the function translate compared to abscissa axe. When m decrease, the center of the function translate to 0. When m increase, the center of the function translate to 1.

If θ is variable and m is constant, we obtain:

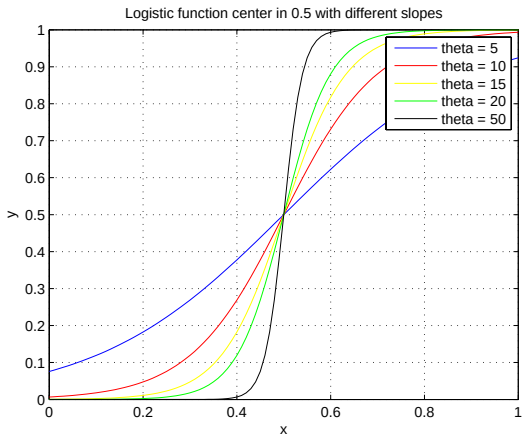

Figure 2: Logistic function for different slopes

We can see that when we change parameter θ, slope of the function increase or dicrease. When θ decrease, the slope of the function decrease whereas when θ increase, the slope of the function increase.

Appendix A.2. presents this function implemented using MATLAB.

In this paper, we will use two logistic functions. Indeed, we must construct a transformation which start near 0 and finish near 1. Hence, we decided to split the transformation in two parts. Each function will have the same parameter m whereas the parameter θ will be different. Parameter m will be determined during analysis histogram while parameter θ will be determined during histogram mapping. Thus, m can be called inflexion point. To illustrate these remarks, we can show an example:

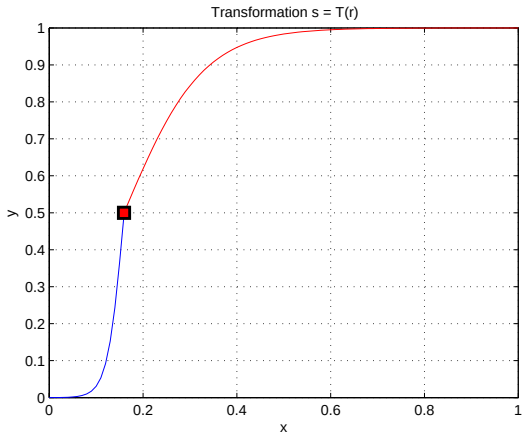

Figure 3: Example of transformation

The point in red correpsonds at the inflexion point dependent on m parameter. Slope of the blue part is defined by θ_1 parameter while slope of the red part is defined by θ_2 parameter.

2.2 Histogram Analysis

2.2.1 Choice and computation of first order statistic

Our method assumes that we must use histogram without use histogram equalisation method. We assume that histogram is described by a Gaussian distribution. The aim of the method is to spread this Gaussian. For this, we must use first order statistics. Both of them are interesting: mean and median. In reality, the distribution is not a perfect Gaussian and can presented some outliers or noise. Unlike median, mean is influenced by noise or outliers and it is not a good parameter to perform the algorithm.

In the section 2.1, we saw that logistic function can be centered using m parameter. In this part, we defined m parameter like the median of the histogram. Definition of the median is as follow:

$$\int_{-\infty}^{m} dF(x)dx \geq \frac{1}{2} and \int_{m}^{+\infty} dF(x)dx \geq \frac{1}{2}$$

where m is the median and $dF(x)$ is the probability density function. Median can be computed using cumulative density function. Cumulative density function is computed as follow:

$$F(x) = \int_{-\infty}^{x} f(x)dx$$

where $f(x)$ is the probability density function. Median can be find as follow:

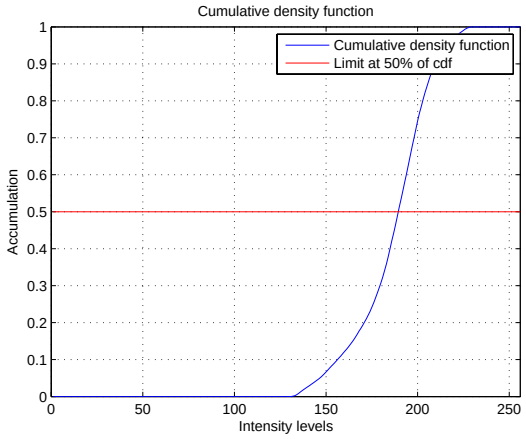

Figure 4: Detection of Median

Median is the intensity level being at the intersection of the cumulative density function and the straight line $x = 0.5$.

Appendix A.1.1. presents the code implemented using MATLAB.

2.2.2 Detection minimum maximum

We assume that the shape of histogram was a Gaussian distribution. Thus, this distribution doesn't use all intensity levels. We can define two parameters: *min* correpsonding the intensity level at 1% of the cumulative density function and *max* corresponding the intensity level at 99% of the cumulative density function. We can illustrate as follow:

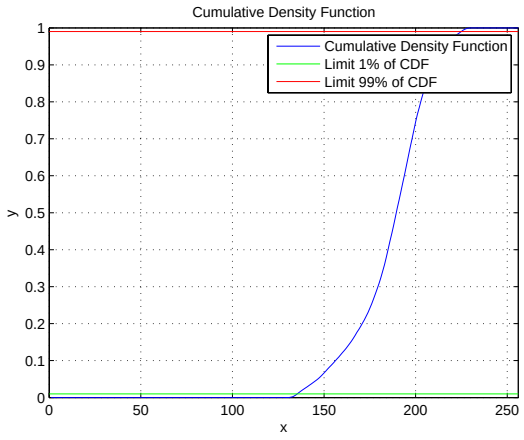

Figure 5: Detection Minimum Maximum

min is the point at the intersection of the line representing limit at 1% and the curve representing the cumulative density function. Similarly, max is the point at the intersection of the line representing limit at 99% and the curve representing the cumulative density function.

Appendix A.1.2. presents the code implemented using MATLAB.

After to have computed median, minimum and maximum parameters, we can perform mapping histogram.

2.3 Histogram Mapping

In this part, we explain how we compute θ_1 and θ_2 previously defined in the section 2.1. To compute this both parameters, we assume that:

$$T(r(min)) \rightarrow \quad 0 \Leftrightarrow T(r(min)) = \quad s(0) \tag{2}$$
$$T(r(median)) = \qquad 0.5 \tag{3}$$
$$T(r(max)) \rightarrow \quad 1 \Leftrightarrow T(r(max)) = \quad s(1) \tag{4}$$

Thus, with equation (1),(2) and (3), we can compute θ_1 parameter whereas with equation (1), (3) and (4), we can compute θ_2 parameter. We obtain as follow:

$$\theta_1 = \lim_{x \to 0} -\frac{\ln \frac{1}{x}}{min - median}$$
$$\theta_2 = \lim_{x \to 1} -\frac{\ln \frac{1}{x}}{max - median}$$

Appendix A.1.3. presents the code implemented using MATLAB.

Now, all parameters allowing to compute the transformation are computed. We obtain the following transformation:

$$T(x) = \frac{1}{1 + e^{-\theta_1 \times (x - median)}}$$

for $x \in [0, m]$

$$T(x) = \frac{1}{1 + e^{-\theta_2 \times (x - median)}}$$

for $x \in [m, 1]$

3 Results and Discussion

3.1 Results

In this section, we will present several tests on different types of images. Configuration of computer to perform tests were:

- Processor: Intel Core 2 Duo T5250 1.50 GHz
- Memory: 4.00 Go DDR2 800 MHz

3.1.1 Girl

We apply the enhancement function and we obtain following result:

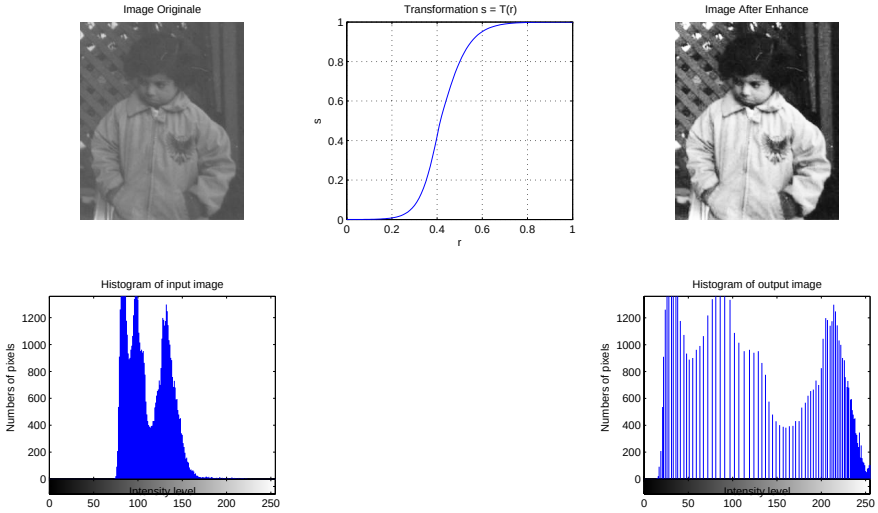

Figure 6: Result for girl image

Size of this image is 291×240 pixels. Computing time last $88,2ms$ therefore the algorithm is fast. On this image, the result is good. The histogram of output image is correctly spread and the quality of the output image is better than the input image.

3.1.2 Fruit

We apply the enhancement function and we obtain following result:

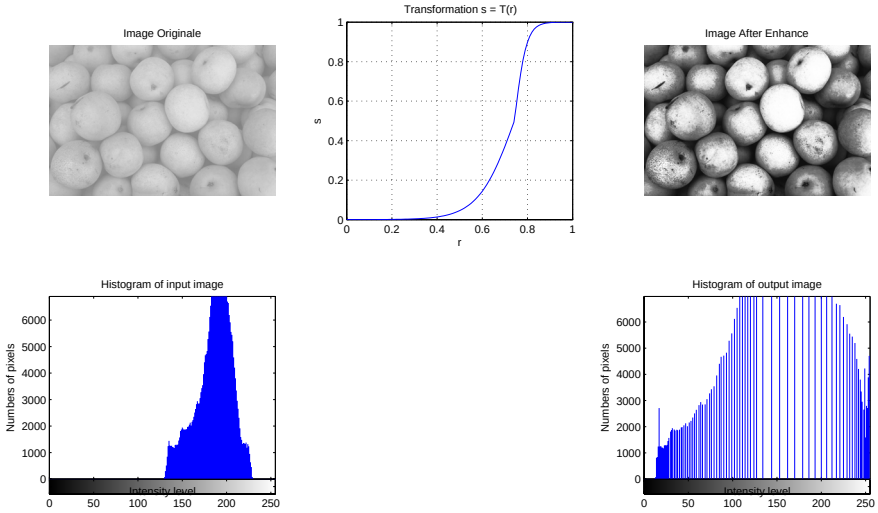

Figure 7: Result for fruit image

Size of this image is 486×732 pixels. Computing time last $425,6ms$ therefore the algorithm is fast compared to the size of the image. Like girl image, the result is good. The histogram of output image is correctly spread and the quality of the output image is better than the input image.

3.1.3 Trees

We apply the enhancement function and we obtain following result:

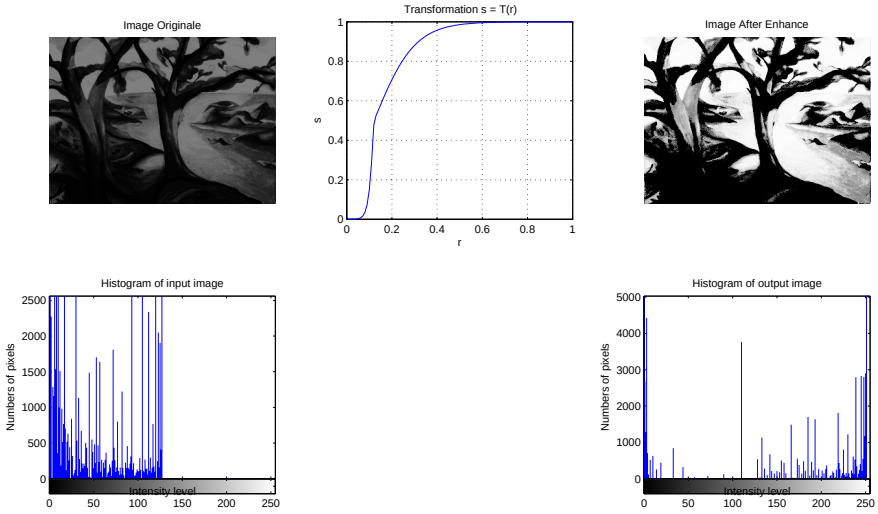

Figure 8: Result for trees image

Size of this image is 258×350 pixels. Computing time last $108,7ms$ therefore the algorithm is fast. Like

two previous images, the result is good. The histogram of output image is correctly spread and the quality of the output image is better than the input image.

3.1.4 Pooh

We apply the enhancement function and we obtain following result:

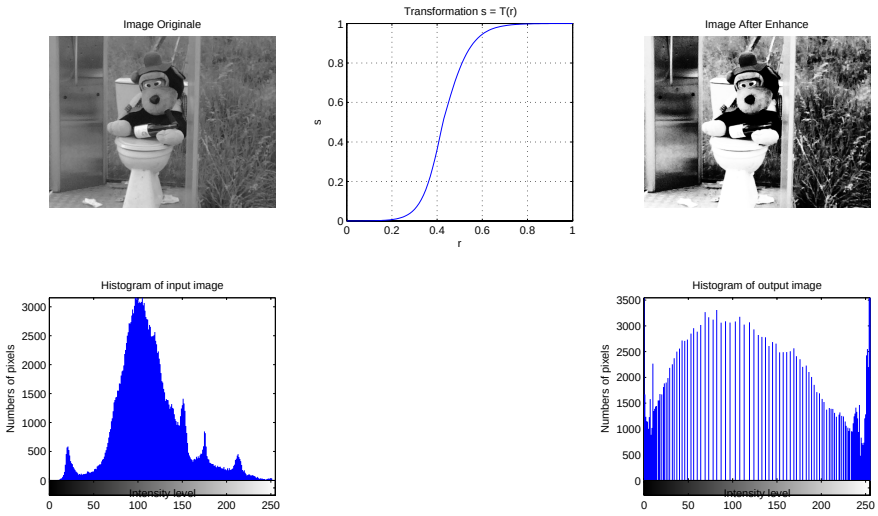

Figure 9: Result for pooh image

Size of this image is 400×528 pixels. Computing time last $267,2ms$ therefore the algorithm is fast compared to the size of the image. The result on this image is not very well. Indeed, we can see that the histogram of the input image is not a Gaussian distribution but a mixture of Gaussian distribution. Thus when we spread the histogram, we tend to delete feature around the median. The consequence can be seen on the output image. The general contrast of the image is better but details are lost.

3.1.5 Saturn

We apply the enhancement function and we obtain following result:

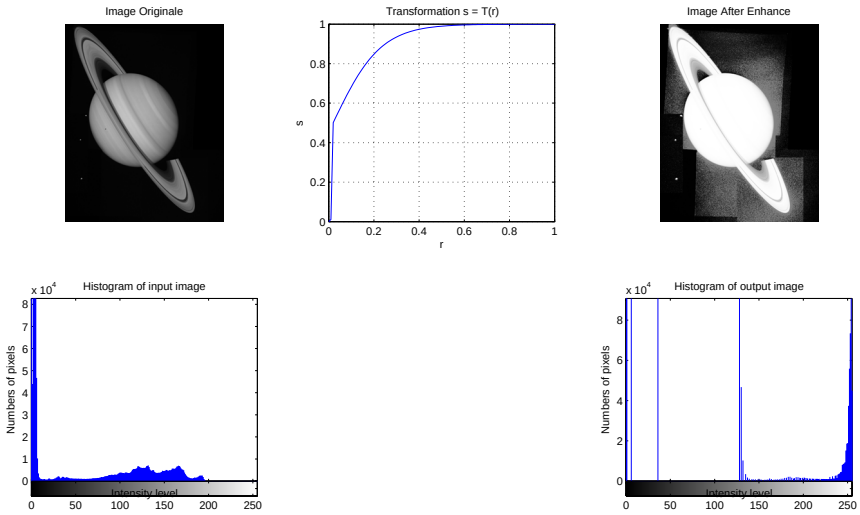

Figure 10: Result for saturn image

Size of this image is 1500×1200 pixels. Computing time last $2,3s$ therefore the algorithm is fast compared to the size of the image. The result on this image is bad. The problem of this image is that histogram follows a Gaussian distribution but with the variance of this Gaussian is very small. But, we spread this Gaussian on the all intensity levels and we produce some outliers.

3.2 Discussion

We saw that results were good when the histogram follow a Gaussian distribution. But in reality, distributions are not always a Gaussian distribution. That's why, we obtained bad results for some images. The better solution is to assume that distributions follow a mixture of Gaussian distribution. Method based on histogram warpring [2] is based on this hypothesis. The principle is to detect valleys around each Gaussian and spread with a transformation based on piecewise of quadratic function. Results will be better because each Gaussian corresponds to feature. Hence, you don't remove contrast between features.

4 Conclusion

This method proposed on this paper is a very simple and fast method to enhance image automatically. We introduced the histogram transformation which composed by piecewise of logistic function. Then, we explain how we can detect parameters allowing to compute the transformation. After that, we explain the mapping histogram to enhance images. Finally, we present and discuss about results obtained.

5 Reference

References

[1] Gonzalez R.C. and Woods R.E., 2007. *Digital Image Processing*, 3 edition Prentice Hall 3.

[2] Grundland M. and Dodgson N. A., 2006. Automatic Contrast Enhancement By Histogram Warping *Computational Imaging and Vision*, Springer.

A Code

A.1 Enhance function

```
%-------------------------------------------------------
%%Function for enhancement of images
%-------------------------------------------------------
function [Image_Out] = Enhance(Image_In)

close all;

%Read input image
imorig = imread(Image_In);

%Check what kind of image with have
%If with have an RGB image, we convert in grayscale
if ndims(imorig) == 3
    imgray = rgb2gray(imorig);
else
    imgray = imorig;
end

%Display original image
figure(1);
subplot(2,3,1);
imshow(imgray);
title('Image Originale');

tic;

%Conversion in double
imgraydouble = im2double(imgray);

%Calculate the histogram of the image
%Number of bins
nb_boxes = 256;

%Size of the images
sizeim = size(imgray);

%Numbers of pixels
nbpixels = sizeim(1).*sizeim(2);

%Calculate histogram
hist = imhist(imgray,nb_boxes);

%Allocation of the cdf
cdf = zeros(nb_boxes,1);

%Compute cdf
for i = 1:nb_boxes
    j = i;
    while j ≥ 1
        cdf(i) = cdf(i) + hist(j);
        j = j - 1;
    end
end

%Compute the median
for i = 1:(nb_boxes)
    if ((cdf(i) > nbpixels/2))
        median = i;
        break;
```

```
    end
end

%Find the moment 1% and 99% of the histogram
%Intensity level at 1% of the cdf
inte1 = 0;
%Intensity level at 1% of the cdf
inte99 = 0;
thresholdleft = 0.01;
thresholdright = 0.99;
for i = 2:nb_boxes
    if (((cdf(i-1)/nbpixels) <= thresholdleft)&&((cdf(i)/nbpixels) > thresholdleft))
        inte1 = cdf(i-1)/nbpixels;
    elseif (((cdf(i-1)/nbpixels) <= thresholdright)&&((cdf(i)/nbpixels) > thresholdright))
        inte99 = cdf(i-1)/nbpixels;
    end
end

%Normalization of the median
mediannorm = median/nb_boxes;

%We must create an transformation with two distinct part
%First is at the left of the median
%Second at the right of the median
%The parameter of the translation is fixed by the value of the median
%For each part with must find the parameter theta to choose the slope of
%the function
%For the left part, we can find theta because we know that y(0)= T(x(inte1))
%= 0 and the median is fixed
%For the right part, we can find theta because we know that y(1)= T(x(inte99))
%= 1 and the median is fixed

%Parameters of left side
thresholdleft = 0.0001;

%Parameters of right side
thresholdright = 0.9999;

%Computation of the slope at the left side of the transformation
valthetaleft = - (log((1./thresholdleft)-1))./(inte1 - mediannorm);

%Computation of the slope at the right side of the transformation
valthetaright = - (log((1./thresholdright)-1))./(inte99 - mediannorm);

%Apply transformation on output image
%Preallocation of the output image
Image_Out = zeros(sizeim(1),sizeim(2));

for i = 1:sizeim(1)
    for j = 1:sizeim(2)
        if imgraydouble(i,j) <= mediannorm
            Image_Out(i,j) = logisticfunction(imgraydouble(i,j),valthetaleft,mediannorm);
        else
            Image_Out(i,j) = logisticfunction(imgraydouble(i,j),valthetaright,mediannorm);
        end
    end
end

time = toc;

%Display the output image
subplot(2,3,3);
imshow(Image_Out);
title('Image After Enhance');
```

```
%Compute and display the transformation
x = 0:0.01:1;

%Preaalocation of y
y = x;
%Size if x
sizex = size(x);

%Calculate the generic tansformation
for i = 1:sizex(2)
    if x(i) ≤ mediannorm
        y(i) = logisticfunction(x(i),valthetaleft,mediannorm);
    else
        y(i) = logisticfunction(x(i),valthetaright,mediannorm);
    end
end


subplot(2,3,2);
plot(x,y);
grid on;
xlabel('r');
ylabel('s');
title('Transformation s = T(r)');

%Compute and display histogram of input and output image
subplot(2,3,4);
imhist(imgray);
title('Histogram of input image');
xlabel('Intensity level');
ylabel('Numbers of pixels');

subplot(2,3,6);
im = Image_Out * 255;
im = uint8(im);
imhist(im);
title('Histogram of output image');
xlabel('Intensity level');
ylabel('Numbers of pixels');
```

A.1.1 Detectiion of median

```
%Check what kind of image with have
%If with have an RGB image, we convert in grayscale
if ndims(imorig) == 3
    imgray = rgb2gray(imorig);
else
    imgray = imorig;
end

%Calculate the histogram of the image
%Number of bins
nb_boxes = 256;

%Size of the images
sizeim = size(imgray);

%Numbers of pixels
nbpixels = sizeim(1).*sizeim(2);

%Calculate histogram
hist = imhist(imgray,nb_boxes);
```

```
%Allocation of the cdf
cdf = zeros(nb_boxes,1);

%Compute cdf
for i = 1:nb_boxes
    j = i;
    while j ≥ 1
        cdf(i) = cdf(i) + hist(j);
        j = j − 1;
    end
end

%Compute the median
for i = 1:(nb_boxes)
    if ((cdf(i) > nbpixels/2))
        median = i;
        break;
    end
end

%Normalization of the median
mediannorm = median/nb_boxes;
```

A.1.2 Detection Minimum Maximum

```
%Find the moment 1% and 99% of the histogram
%Intensity level at 1% of the cdf
inte1 = 0;
%Intensity level at 99% of the cdf
inte99 = 0;
thresholdleft = 0.01;
thresholdright = 0.99;
for i = 2:nb_boxes
    if (((cdf(i−1)/nbpixels) ≤ thresholdleft)&&((cdf(i)/nbpixels) > thresholdleft))
        inte1 = cdf(i−1)/nbpixels;
    elseif (((cdf(i−1)/nbpixels) ≤ thresholdright)&&((cdf(i)/nbpixels) > thresholdright))
        inte99 = cdf(i−1)/nbpixels;
    end
end
```

A.1.3 Computation of θ_1 and θ_2

```
%We must create an transformation with two distinct part
%First is at the left of the median
%Second at the right of the median
%The parameter of the translation is fixed by the value of the median
%For each part with must find the parameter theta to choose the slope of
%the function
%For the left part, we can find theta because we know that y(0)= T(x(inte1))
%= 0 and the median is fixed
%For the right part, we can find theta because we know that y(1)= T(x(inte99))
%= 1 and the median is fixed

%Parameters of left side
thresholdleft = 0.0001;

%Parameters of right side
thresholdright = 0.9999;

%Computation of the slope at the left side of the transformation
valthetaleft = − (log((1./thresholdleft)−1))./(inte1 − mediannorm);
```

```
%Computation of the slope at the right side of the transformation
valthetaright = - (log((1./thresholdright)-1))./(inte99 - mediannorm);
```

A.2 Logistic function

```
%Create logistic function
function [P] = logisticfunction(x,theta,transx)

%Define logistic function
P = 1 ./ (1 + exp(-(theta * (x - transx))));
```