# CS6640: Image Processing
# Project 4
# Hough Transform

Arthur Coste: *coste.arthur@gmail.com*

November 2012

# Contents

## 1 Introduction

The objective of this fourth project is to implement and study Hough Transform. Indeed, as we have already studied in Project 2, extracting feature from images an mostly edge is an important and challenging objective of image processing. In this assignment we will study how Hough transform is aiming at extracting these edge features from various objects of an image. In this report we will cover the theoretical presentation of the Hough Transform and some interesting points related to its implementation. We will then provide results that we will present and analyse. We will try as much as possible to compare the different methods and solutions we came up with. This report will firstly present the Hough transform without considering the edge orientation and then considering it, so we will compare both in a last part. We will finally end this report by showing the implementation of the methods we discussed. Some of them were already implemented in previous assignments so we are going to remind them briefly.

The implementation is also realized using MATLAB, and here are the related functions for this project.

Note: The following MATLAB functions are associated to this work:

- $select\_points.m : [x, y] = select\_points(InputImage)$
- $Hough\_transform\_1.m : ReconstructedLinesImage = Hough\_transform\_1(decrement, filtering)$
- $Hough\_transform\_2.m : ReconstructedLinesImage = Hough\_transform\_2(filtering)$

Note: I used images that were provided and made all the other drawing or images so there is no Copyright infringement in this work.

Note: Due to the implementation of Hough transform I made and the structure of images, thresholding has been set up to be performed only on certain images. So when running the script, you don't have the choice of images because I set up thresholds to provides results according to certain images.

# 2 Theoretical presentation

In this first part, we are going to introduce and present Hough transform, the theory and the main characteristics.

## 2.1 Hough Transform

Hough Transform is a technique invented by Paul Hough in 1962 to extract edge features from an image. Hough transform can be described as a mapping function which convert a point of the Image space into a line or a curve in Hough Space. Then, using some properties of Hough space we can detect and identify some groups of pixels that share common properties such as being on a same line or crossing set of lines, which will provide us the information required to draw these lines.

### 2.1.1 Image space parametrization

The image space is the space of all pixels belonging to a random input image where we want to extract some features. A preprocessing step will be necessary to remove noise which will cause issues. In this part, we will assume that we are in an ideal case with no noise.

In the image space, we work in a discrete raster defined with the Direct Orthonormal Cartesian bi-dimensional space basis composed of the two vectors :

$$\mathbf{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad \mathbf{e_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1}$$

So in such a basis a line or an edge is defined with the following affine equation:

$$y = ax + b \qquad \text{with} \begin{cases} a = \frac{\Delta y}{\Delta x} \text{ being the slope of the line} \\ b \text{ being the y intercept} \end{cases} \tag{2}$$

In the previous equation, the position values in the basis : $(x, y) \in \mathbb{R}^2$ are defined and well known, but we ignore the values of the slope and the y-intercept of the line or edge between points. The point of using Hough Transform is to create a new space where these two unknowns can be determined.

### 2.1.2 Hough space

Hough space is a parametric hybrid space. Indeed, it's linked to the image but has no physical reality compared to it. Hough space is composed of two parameters computed using the Polar bi-dimensional space basis composed of the two polar vectors $(e_r, e_\theta)$. We can of course express our image position coordinates into this basis.

$$\begin{cases} x = rcos(\theta) \\ x = rsin(\theta) \end{cases} \qquad \text{with} \qquad \begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = arctan(\frac{x}{y}), (x, y) \in \mathbb{R}^{+2} \end{cases} \tag{3}$$

Due to the discrete structure and the fact that we are using MATLAB numerical indexation of arrays we will restrain the definition to strictly positives values of position coordinates.

So we are going to associate to each point in the image space that we consider interesting or that belong to an edge a sinusoid curve in Hough space using the following transformation:

$$\rho = xcos(\theta) + ysin(\theta) \tag{4}$$

#### 2.1.3 Illustrations

Hough transform has several interesting properties, even if in this case we restrain ourselves only to straight lines. We are going to present, through examples we realized these properties and illustrate what we presented before. So as we said before Hough transform maps a point of the image space into a sinusoid curve in the Hough parametric space $(\rho, \theta)$.

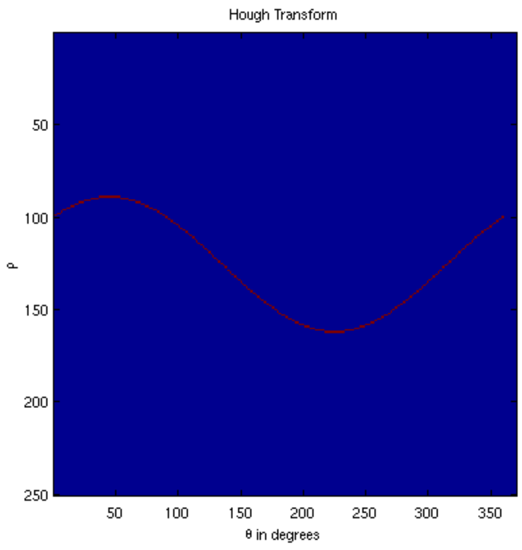

Figure 1: Sinusoid in Hough parametric space associated to a point in the image space

The shape of the sinusoid is defined according to the position of the point regarding the origin of the image from which the parameters are computed. Indeed if the point is far from the origin, the curve is going to have a wider amplitude. On the other hand, the closer the point gets to the origin the smaller become the amplitude. The position of the origin is an important aspect. Indeed if we let the origin at $(0, 0)$ points really far are going to have a very large amplitude, and we can have a bigger Hough space. So in the previous picture and so is in the following, we moved the origin to the middle of the vertical axis to have a centred sinusoid display.

Once we know that a point is mapped into a sinusoid curve we can easily do that for two points. And as a matter of fact a very interesting property is going to be revealed because two points are always aligned in the Image space.

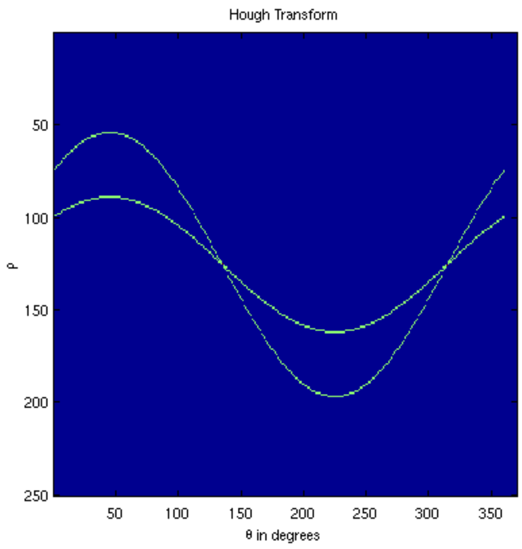

Figure 2: Sinusoid in Hough parametric space associated to two points in the image space

So as we can see on the previous picture, the two points are mapped into two curves that cross each other twice due to the periodicity of the sinusoid functions. As we are working in Hough parametric space, the coordinates are $\theta$ on the horizontal direction which vary from 0 to 360°. The most important point of Hough transform is hidden in the previous picture. Indeed, as we said the curves cross each other, and this point has two coordinates in the Hough space which represent the slope and the y-intercept that describe the line which connect the two points in the image space. This property is going to be used later and generalize to a larger set of points composing edges or lines but this is one of the fundamental properties of Hough transform.

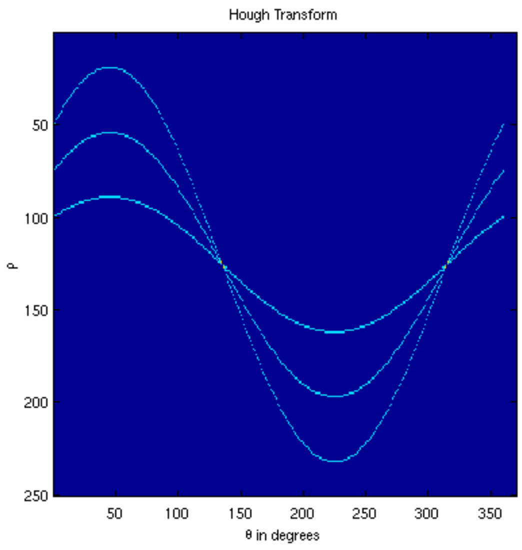

Figure 3: Sinusoid in Hough parametric space associated to three points in the image space

Let's know present how we get back to the image space to create the line. Indeed, as we have seen, thanks to this intersection point, we are able to determine the set $(\rho_l, \theta_l)$ that define the line. So using this information we can compute the line in the image space using:

$$y = \frac{-cos(\theta_l)}{sin(\theta_l)}x + \frac{\rho_l}{sin(\theta_l)} \tag{5}$$

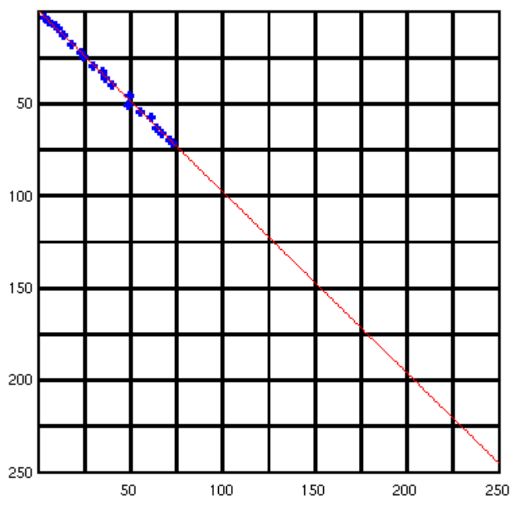

Figure 4: Image space line associated to our maximum parameters in Hough space

As we can see the line perfectly matches the set of points we plotted to simulate an edge. We can use this discussion to show another property of the Hough Transform which is robustness regarding outliers. In fact because the line parameters are computed based on the ”maximum score” (vote strategy described in the implementation section) it’s pretty insensitive to outliers. In our ideal case, we can simulate the presence of noise creating outliers around our simulated edge.

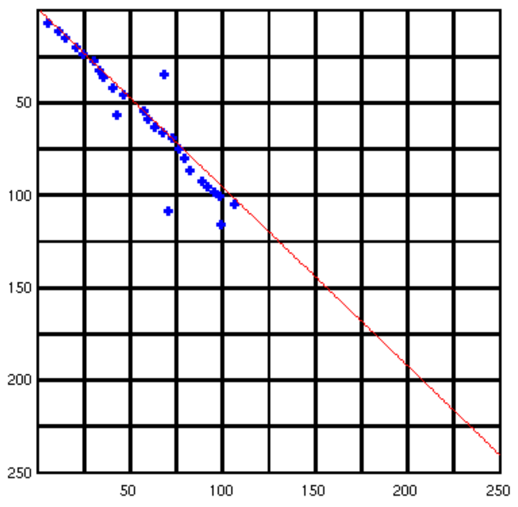

Figure 5: Outliers influence

As we can see if we have quite few outliers, this technique is fairly robust. But if the curves appears to be defined by a "divergent" set of point the Hough Transform won't be able to figure the line. I mean by divergent that there are more points considered around the line than points really belonging to a close neighbourhood of it.

Another concern while looking at the previous picture is that having the slope and the y intercept is not enough to define an edge. Indeed what we have is a line which is by definition infinite while our edge is completely finite. So the challenge here will be to determine the limits of that edge. In our case, due to the ideal nature of our example we can constrain the line between the first and the last point defining the line. Which is the point with the lowest x-value and the biggest x-value.

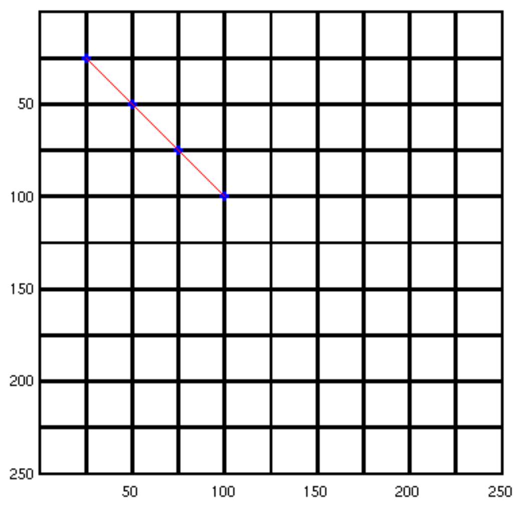

Figure 6: Final line with constrained length

And as we can see in this ideal case, we really have the line defined by the points we used. This presentation was giving the basis of Hough transform in an ideal case, where we know and control all aspects of the transform implementation, but as we are going to introduce and present later in this document, it is harder than it appears previously.

Indeed, one of the challenge is the exact determination of the parameters of the lines which relies on determining the point where most line curves and which is has the maximum score in Hough parameter space.

## 2.2 Improvement

As we presented before the detection of maxima scores of the accumulator image is a serious challenge. It reminds of a previous issue we had in a previous assignment where we had to find a nice threshold value in a correlation image to locate the position of a template in an image. Here the situation is pretty similar, indeed, we are looking for the location of maxima in an image with regard to the surrounding area and false positive maxima. An illustration of this proximity between the two situation is presented in the implementation section.

### 2.2.1 Image filtering

By using a Gaussian kernel as a filter to smooth the image we intend to create an image where we reduce noise. Indeed, this will create a closely continuous evolution of values in Hough space. By doing so we hope to be able to have a better detection of the local maxima in the image. Everything done in this section rely on what has been presented about filtering and smoothing in project 2.

### 2.2.2 Decremental strategy

A solution to improve the accuracy to detect the most suitable values for the slope and the y-intercept associated to each aligned set of points is to use the method presented by Gerig *et al.* The idea here is to apply a second pass on the Hough space to remove a lot of useless parts of the accumulator and preserve the local maxima that carry the information regarding the lines of the original input image.

## 2.3 Two interesting functions

Another interesting point to make is related to the analysis of Hough parametric space. Indeed we want to analyse it to determine and characterize the composition and the structure of the image and then we want to extract some features of interest which happens to be, due to the incremental technique, maxima of the image.

To perform the first task, analyse and characterise in Hough space we need to be able to distinguish patterns. To do so we can rely on the use of the logarithm function which allow us to convert a wide variation of values into a smaller one. This technique allow us to have a clearer representation of the the parametric space. Here are some examples illustrating the previous statement.

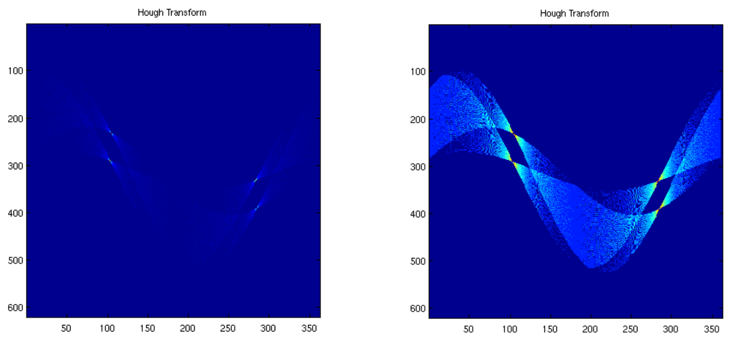

Figure 7: Comparison between regular Hough space and the log space

This aspect is only to be used as visual exploration, because it create a wider range of maximal values than it should.

To perform the maximum detection, an other mathematical function can be used but in this case to produce the opposite effect. Indeed in this case we want to be able to separate values from each other and enhance contrast. To perform this task we can rely on the use of the exponential function, which will convert a small variation into a big variation, which can be really powerful to increase the variation between points and extract easily a set of maximum values using thresholding in the parametric space. In the case of two lines in an image, as presented in the previous results of Hough space, if we apply an exponential function on the regular space, we obtain two white dots, representing the stronger line, and if we lower a bit the threshold we can easily extract the 4 maximum peaks presented before, because the difference with their neighbours is increased a lot so they can be easily separated.

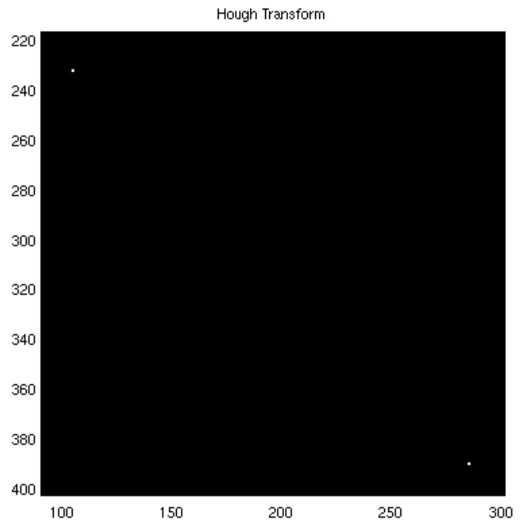

Figure 8: exponential function of Hough space

This procedure is just a simple way to increase the difference between points and try to make relevant maxima pop out.

# 3 Hough Transform on images containing straight lines

## 3.1 Image Preprocessing

As we mentioned in the theoretical section, the goal of Hough Transform is in our case to detect lines or edges. To do it in good conditions we need to use images where edges are pretty well defined. This step is an important and required step for both methods we are going to study in this project. So binary images obtained using simple thresholding can be excellent candidates. Another interesting approach to use on more complex images where objects cannot be easily segmented using a single threshold is to use a technique we already implemented in the second project: edge detection. We will quickly introduce these two methods and how they work. The focus of this document being on Hough transform, this step is just supporting it.

### 3.1.1 Thresholding

Thresholding is a simple image processing technique used to create binary images from a grayscale image. We determine a threshold value and all the values below this threshold will be considered as background and all the values above will be considered as being objects. Thresholding transfer function is a step function where the step occurs at the threshold value. Here is a result of thresholding an image that we realized in project 1:

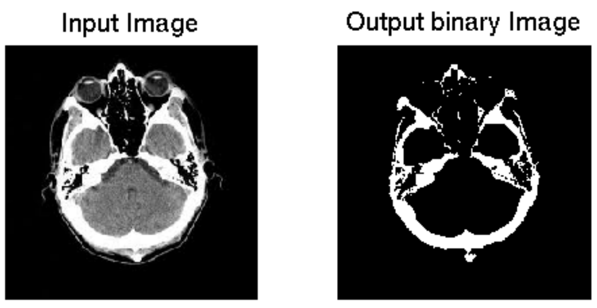

Figure 9: Threshold image of a brain CTscan

The previous image is here to illustrate the thresholding technique, due to its non linear shape it's not a good candidate for our line detection Hough transform.

### 3.1.2 Edge detection

Edge detection is also an image processing technique used to identify areas where the gray level has a strong and quick variation. The simple version of this technique relies on the computation of the image gradient. To do so we rely on filtering techniques with a first or second order discrete derivation kernel. More advanced technique exists and can produce more accurate and sharp results such as the Canny method. Here is an example of edge detection using Sobel's filter that we realized in project 2.

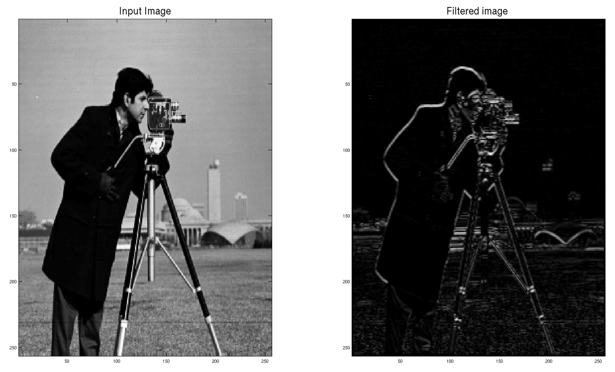

Figure 10: Cameraman image filtered with a Sobel kernel of dimension 3

The previous image is also illustrating a result of edge filtering on an image, it is also not one of the best candidate to perform our line detection using Hough transform.

### 3.1.3 Home-made binary images

We can also realize ourself some test images to present some interesting properties of Hough Transform.

The first example is using a square and show the associated hough space:

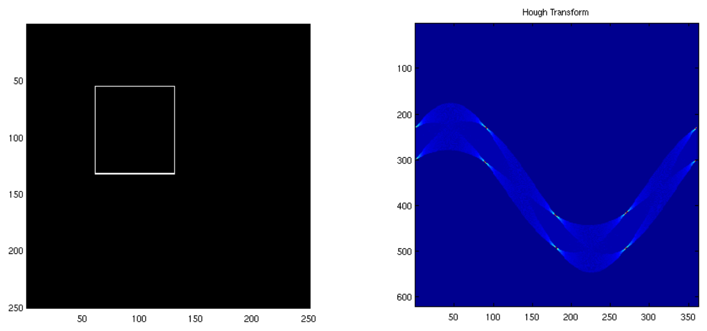

Figure 11: A rectangle and its associated Hough space

As we know, a rectangle is composed of four lines segments with two pairs of parallel segments and pairs of orthogonal lines. These aspects can be found in the previous picture of Hough space where we can see the parallel lines with two sets of maxima lying on top of each other at a constant angular parameter. Due to the fact that in our previous source image the lines are perfectly lying on the vertical and horizontal axis of the image it does make sense to have the angular parameter value equal to 0° and 90°. They correspond to the two first maxima sets of maxima on the hough image. Then we also have maxima at 180° and 270° which are copies of the two previous sets due to the mathematical properties of sinusoidal functions as being periodic with a period of 180°.

Here is an other example of a common shape: a circle:

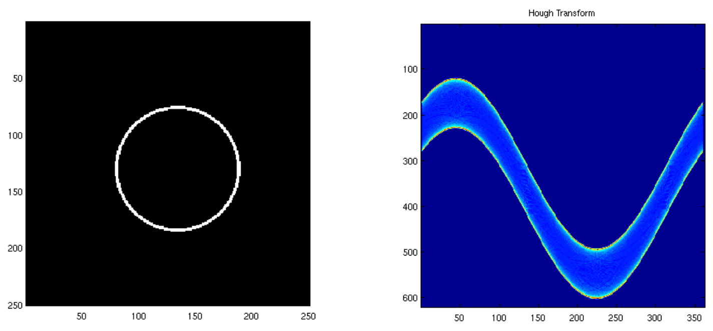

Figure 12: A circle and its associated Hough space

This previous image shows Hough space associated to a circle in the image space. As we know, a circle can be defined with an infinite number of lines tangent to all of the points composing the circle. This is why when computing the associated Hough space, we obtain a set of parallel sinusoid that never cross. Indeed if we worked more on that image we could maybe extract one information, the diameter of the circle which separate a set of parallel curves associated to two points of the circled opposed to each other.

The results we presented before were made with a strong focus on the study and presentation of Hough Space with no intentions to detect points and back map them into lines in the input image. Indeed, as we are going to see and as you may have noticed Hough space pictures are bigger than expected and some modifications have been applied to produce the best results in this space.

## 3.2 Without edge orientation

In this first part we are going to present and analyse the results of our implementation when applied on some specially made images or on some images that were provided.

### 3.2.1 Applying Hough Transform

In this part, we are focusing on applying Hough transform on images. We will in a first step compute the associated Hough space as presented before and then compute the lines provided by the analysis of this space. In a first part we are going to work on some of the images presented before and some binary home-made images. This will allow us to work with images with special characteristics.

Let's apply it to a simple binary image composed of two almost parallel lines. Here is our original input image:

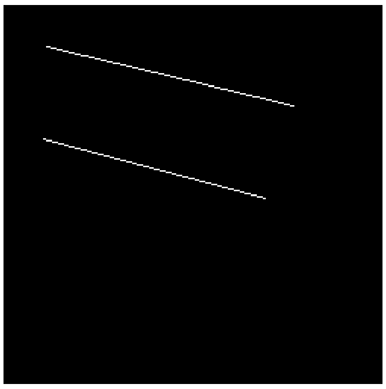

Figure 13: Two lines test image

We then compute the Hough space associated to the pixels where the intensity value is not equal to 0. We obtain the following result using the log function and a :

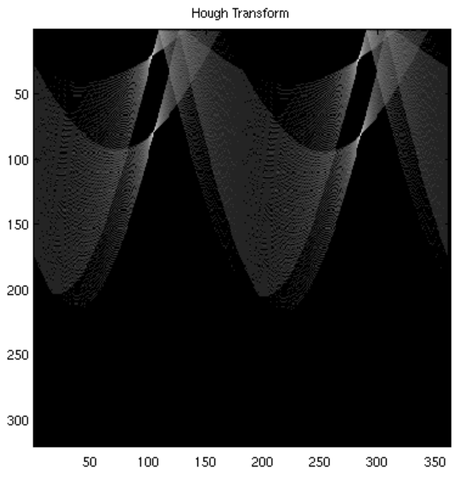

Figure 14: Hough space associated

As we can see here, I used the logarithm function to enhance the contrast on the Hough space image. We can also notice that we don't get the sinusoid shapes we have already see on the previous images. It's due to the fact that I previously was shifting the images so I can have a nice and centred display. But the shape remains the same with two sets of almost vertically aligned points representing our two parallel lines.
In a first time if we just threshold Hough space to extract the maxima we can already obtain some interesting result. Indeed, if we just extract the maxima value and plot the line here is the result:

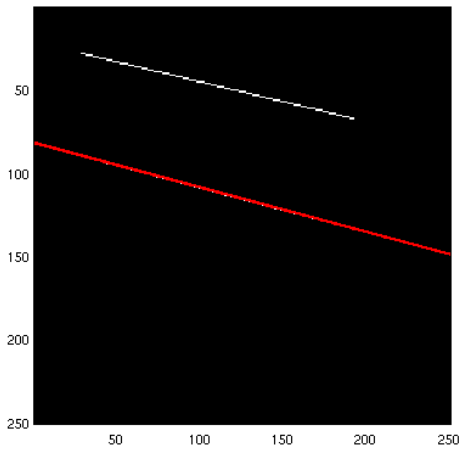

Figure 15: Reconstruction of the line associated to the maximum value

If we lower the threshold value we can then detect the other line but as we can also see on the following picture we detect another line which is absolutely not representing a line of our original input image. We think that this line is a repetition of one of the previous, because of the common slope, but the shifted.

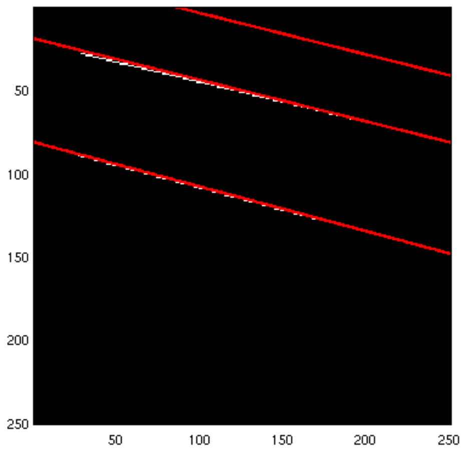

Figure 16: Reconstruction of the lines of the original input image

This other line is due to the periodic behaviour of sinus functions we described before. The shift is due to the y-intercept of the line which is computed using the angular parameter. So if we limit ourselves to display only lines whose angular parameter lies in $[0°, 180°]$ we finally obtain what we were looking for:

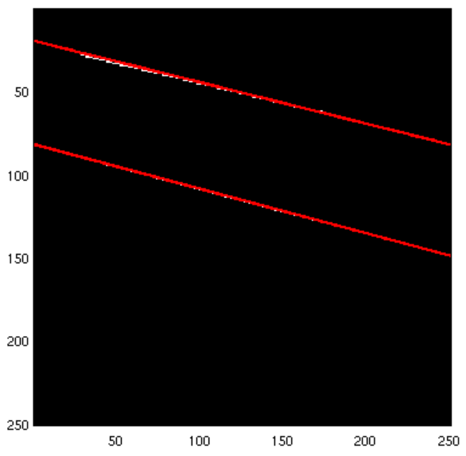

Figure 17: Reconstruction of the lines of the original input image

Now that we have shown in a very simple image that our implementation of Hough transform was producing the expected results we are going to work on more complicated images, with more lines with different slopes. In a first time we are again going to work on an home made test image. It's still a quite simple image but as we will discuss later it could be a result coming from some processing techniques.

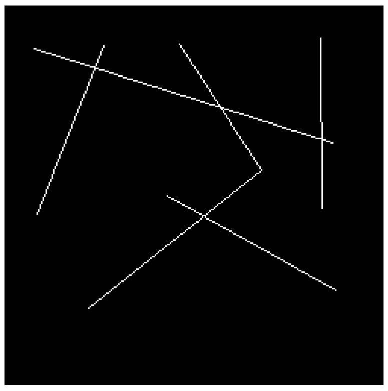

Figure 18: Many lines test image

Here is the associated computed Hough space:

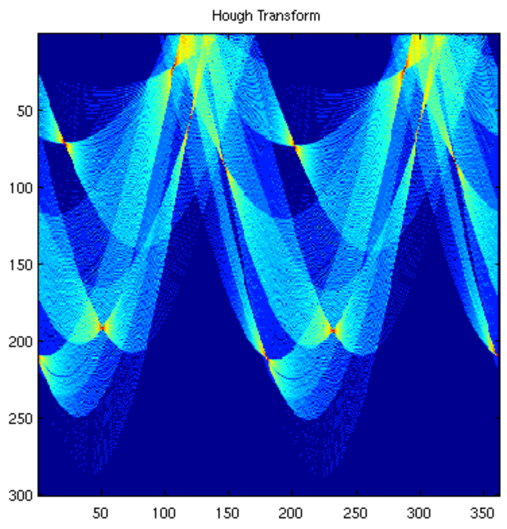

Figure 19: Log Hough space associated to our lines test image

As we can see on the previous images, if we look on the left half of the image, we have 6 appearing areas of reddish color, showing that we have a maximum lying in the area. The challenge here is that all maxima does not have the same intensity so we will have to threshold according to a lower value to get all of them, but this could lead to detect more lines than what we expect. Let's see what we obtain:

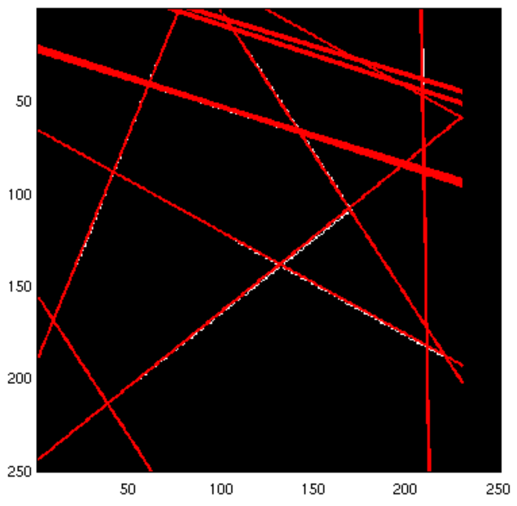

Figure 20: Reconstruction of the lines of the original input image

As we can see, our threshold value was low enough to detect at least all the lines of the input image. But as we can also see, there are some other "undesirable" lines due to an "over detection". Having these lines is due, in my opinion, to the thresholding technique which keep some points of high value and create thicker lines when back mapping them. Then we also have the shifted lines we had before, because in this case to be able to have all the lines we need to cover the whole set of theta values. This comes probably from our implementation which is not optimal and has some issues to manage vertical lines, whose angle is set to 360°. So the result is a bit cluttered, but we will try to see if we can get rid of those drawbacks using the decremental technique.

Finally, before discussing the implementation of the decremental technique, we are going to apply our Hough transform to real images that we preprocessed before. This image has been provided for this project, we processed it to extract the edges.

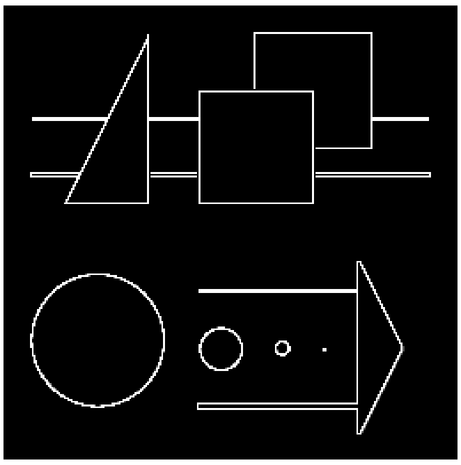
Figure 21: Processed image

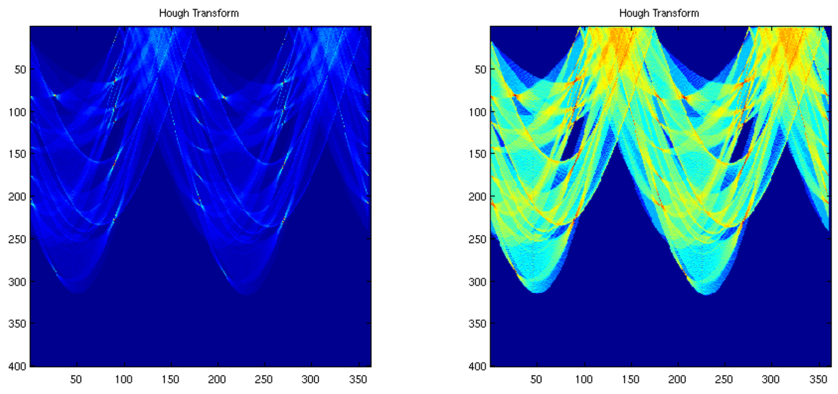

Figure 22: Regular and Log Hough space associated to the image

As we can see, the associated log Hough space is pretty cluttered and complicated to analyse, this is why we also plotted the regular space. This image is really challenging because as we can see on the two images above there are only few maxima pretty well defined, some of them are less strong and some could be lost in the yellowish reddish area of the top of the picture. This is why it's complicated to have a complete and nice detection of all the edges of the image, mostly due to the use of this basic threshold method. Here are some results.

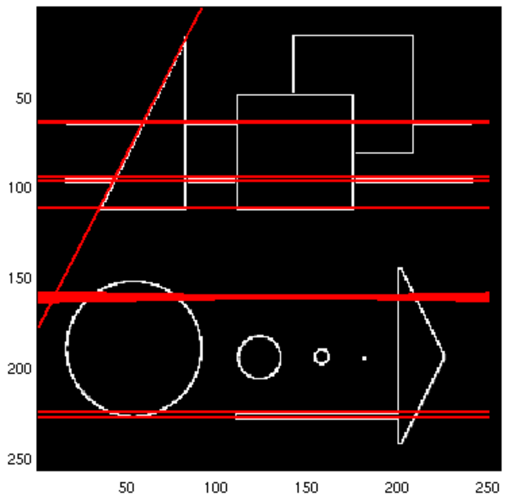

Figure 23: Reconstruction of the lines of the original input image

If we lower the threshold value we detect a larger number of lines and increase the redundancy because we have several computed lines associated to a single edge, due to the detection of neighbouring pixel of a maximum value.

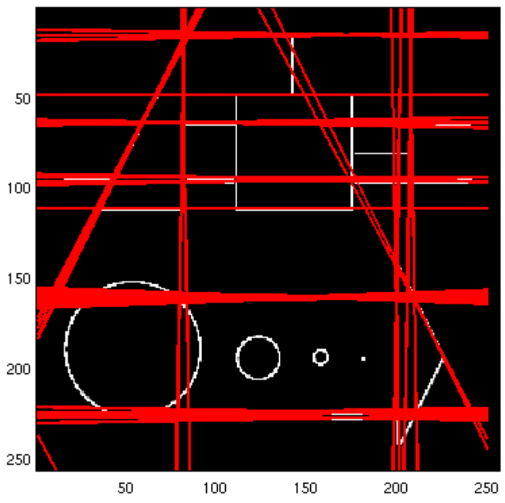

Figure 24: Reconstruction of the lines of the original input image

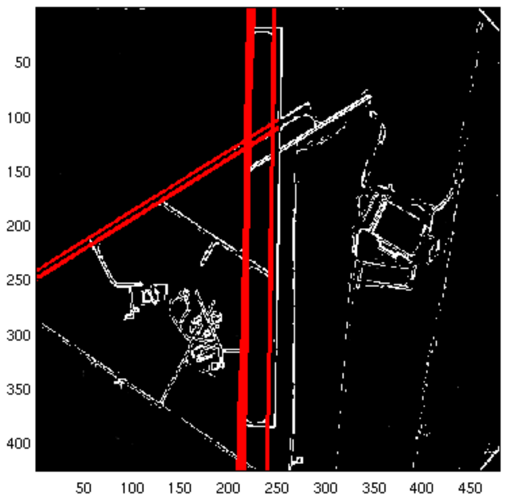

Figure 25: An other example of principal lines detection

### 3.2.2 Using decremental strategy

I don't consider being a specialist, I'm just curious and I wanted to try to implement it to see how it influences the results. So my implementation and results might not be the most accurate and nice. If we apply a second pass in the Hough space where we decrement points that are different from the maximum value as presented before, we are going to remove pixel of low values. Indeed this second pass will allow us to keep only the maximum intensity areas which will hopefully help us to locate and determine the coordinates of our maxima. Although this second pass reduces significantly the amount of information and keep only maximal values areas, we still need to apply a threshold to the values we consider to avoid having too much lines computed from remaining pixels. We apply this strategy to our generated image.

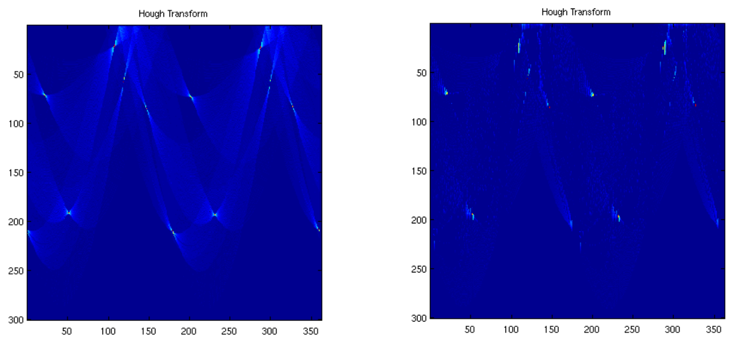

Figure 26: Comparison between original and decremented Hough space

As we can see on the previous image, compared to the original hough space the amount of information displayed has been reduced. But there is still a challenge to detect peaks. Indeed peaks are proportional to the line length, the longer the line, the higher the peak because more lines will go through a small neighbourhood. So detecting long lines will not be too challenging because they will have the higher value for their peak. Regarding small line we still have some issues, this is why the post decremental method thresholding is aiming at capturing those points associated to smaller lines.

If we apply it to the shape images we can also extract pretty well images even if my implementation might be wrong.

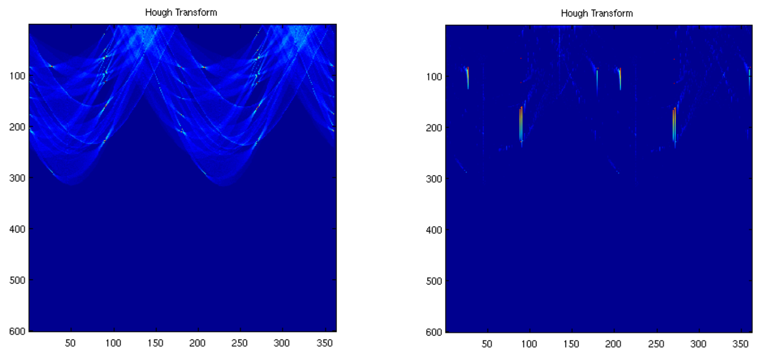

Figure 27: Comparison between original and decremented Hough space for shapes image

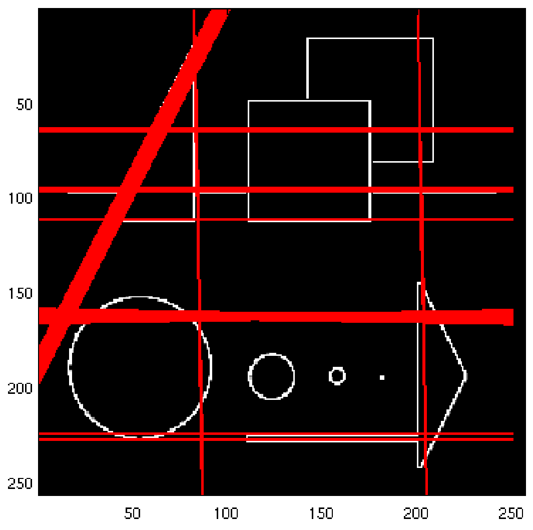

Figure 28: Lines reconstruction using the decremented Hough Space

As we can see, it seems to be working but there might be some aspects to improve to have a more accurate result.

### 3.2.3 Using smoothing of Hough Space

An other approach suggested in the project was to perform a Gaussian smoothing of Hough space before performing peak detection. This technique is also working pretty well. Here are some results:

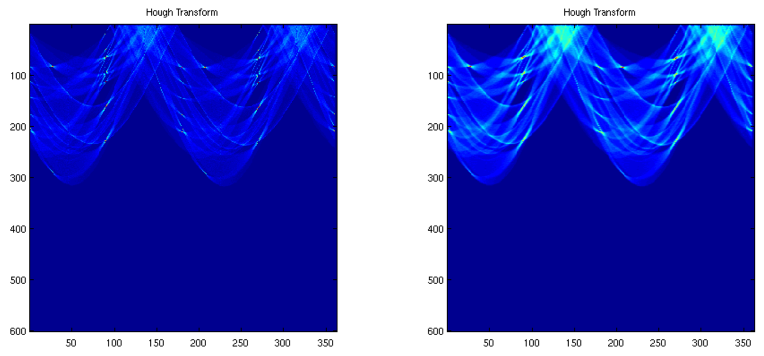

Figure 29: Comparison between original and smoothed Hough space for shapes image

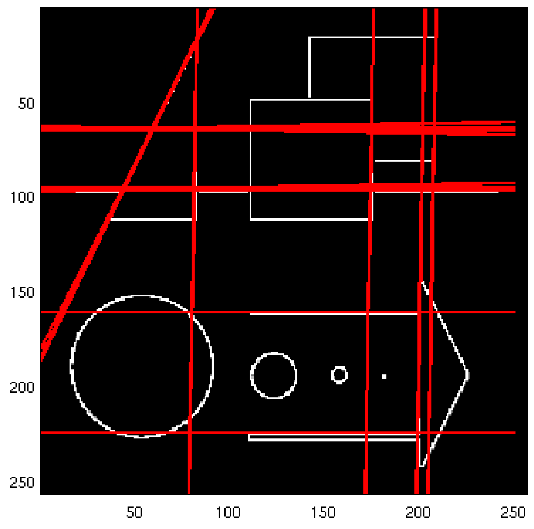

Figure 30: Lines reconstruction using the smoothed Hough Space

### 3.2.4 Other examples

While testing our implementation of Hough transform some concerned appeared. Indeed, the influence of two aspects appear to be important: the length of the edges and their thickness. Indeed the larger are the two previous parameters the larger became the score in Hough space. So I wanted to look deeper at those two previous aspects. I created two test images and use them in the pipeline we have described before.

Firstly let's take a look at length influence. Indeed the longer the line the more pixel we have so the higher should be the peak. Let's see :

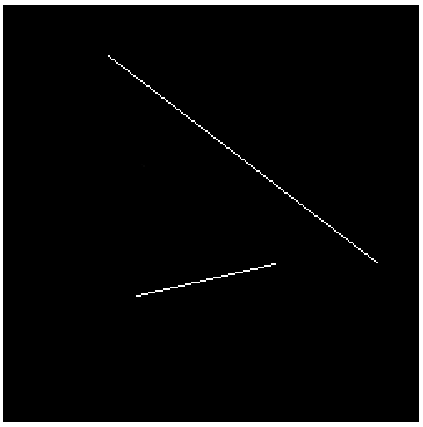

Figure 31: length test image

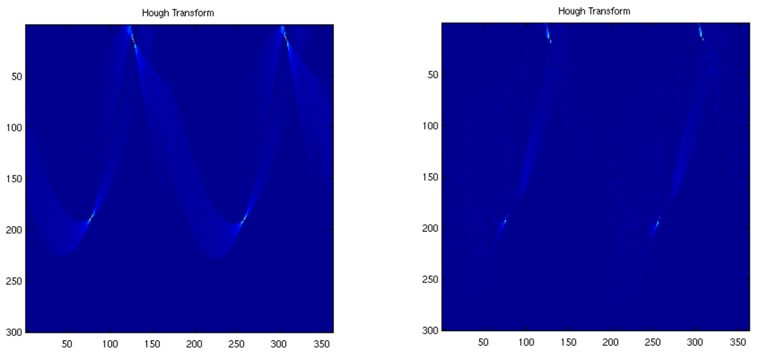

Figure 32: associated Hough space and decremented Hough space

As we can see, the filtered Hough space show two main areas, but as we predicted the only line which is detected with a small threshold is in fact the longest line.

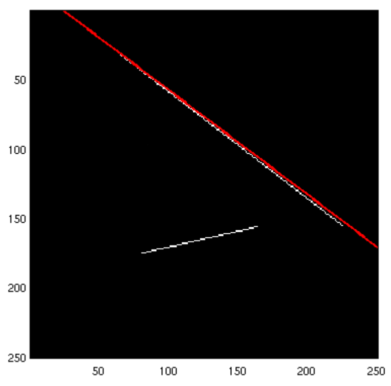

Figure 33: Result

That's maybe not an outstanding point but, it allows us to have a clearer interpretation of the previous results we showed. And by the same process of the number of points composing lines we obtain the exact same conclusion regarding the influence of the thickness of the line. Indeed this has some consequences on the thresholding performed to extract other edges whose associated maximum could be far lower than the absolute maximum associated to the most important edge. So this is why a pre processing of the image might be necessary and maybe stronger than just an edge detection with regard to the structures we care about and what we are looking for as a result.

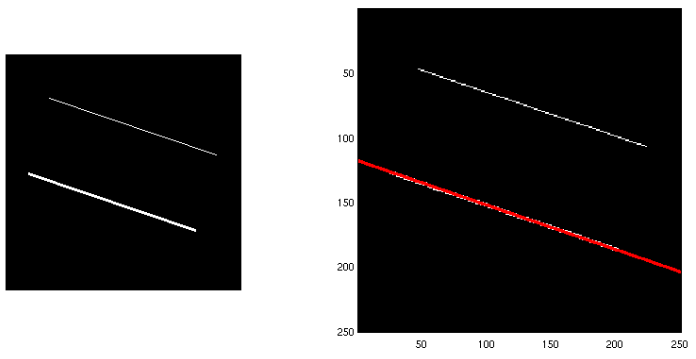

Figure 34: Thickness comparison result

### 3.3 Using edge orientation

In this section, we are going to improve the pipeline we presented before in order to reduce computation time. Indeed we saw that computing the Hough space associated to an image is a relatively long operation, and could become longer if we combine it with the decremental technique for instance. And as we know the computational expense of an algorithm might be a critical aspect in some situation so looking to solutions to reduce it is always an interesting aspect.

#### 3.3.1 Preprocessing

For the same reasons as presented before an edge detection pre processing is necessary on our input images. But in order to improve what has been presented so far, we are also going to improve this step. Indeed as we saw in project 2, by using a derivative based filter, it's possible to extract the gradient of an image and then compute its magnitude and orientation. At that time when doing project 2, we assumed that the gradient orientation map was aiming at showing areas of constant orientations and help to locate the discontinuities in the image. Now we are going to pre process this image for the same reasons as we explained before and then use it to only accumulate a reduced set of points instead of a full curve in Hough space.

Here is a result of this method applied to the Ohio runway picture provided for this work:

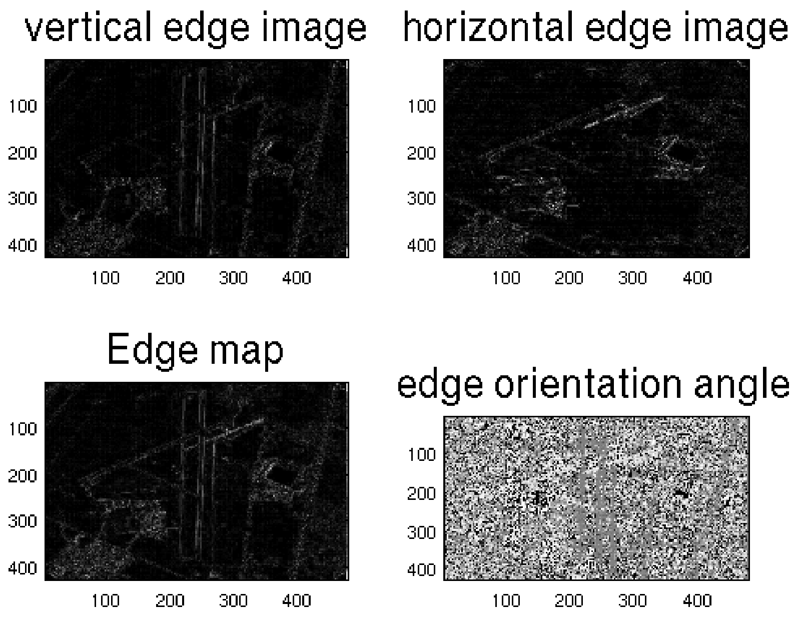

Figure 35: Gradient of the image, magnitude and angle maps

If we look at the angle map we can distinguished the straight lines representing the runway where the gradient angle stays the same because the gray level stays almost the same. We need to process this image to extract two different kinds of gray levels representing the two different set of lines composing the runway. The following image is representing presenting our angle map where the angle belong to $[-180°, 180°]$ and its associated absolute value map.

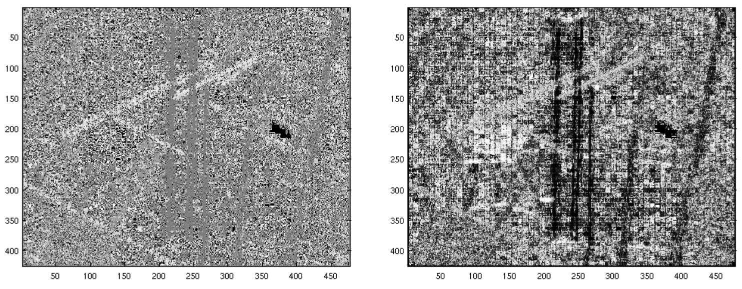

Figure 36: angle map and it's absolute value

Now we can obviously say that the local orientation of the vertical line is associated to an angle of $0°$ but we can mostly know that the other runway have an approximate angle of about $60°$. But one important issue is that this image is really really noisy so really hard to analyse. We are going to use this image with the gradient magnitude map in our pipeline to get for a given point on an edge, the local value of the angle associated to it. Due to the noisy behaviour we described before we will need to associate a tolerance to that angular value we get from the image. We also need to convert the gray level contained in our angle map into a meaningful angular value in the previous interval : $[-180°, 180°]$. To do so we use the following linear transformation (scaling):

$$\theta(i,j) = \frac{I(i,j) \times 360°}{255} \tag{6}$$

And finally before applying our pipeline we need to define a tolerance interval on the previous computed angular value. On one hand, due to the noisy structure of the image it seems safe not to consider a too small interval but on the other hand if this interval is too big we will end up with the same issues has we had before with too many pixels of high score and without being able to separate them. We are going to start with a $\pm 5°$ interval around the value read in the angle map. Here are some results:

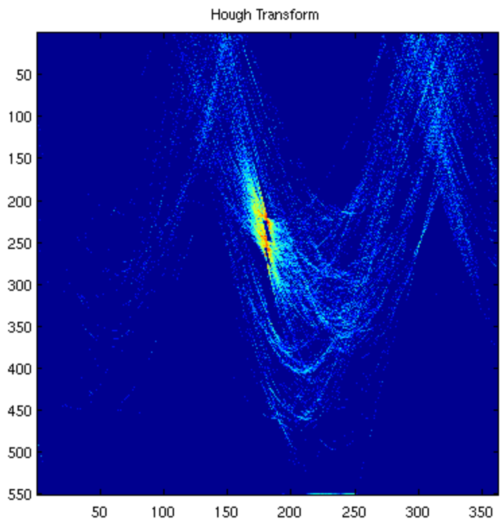

Figure 37: Log Hough parametric space using edge orientation

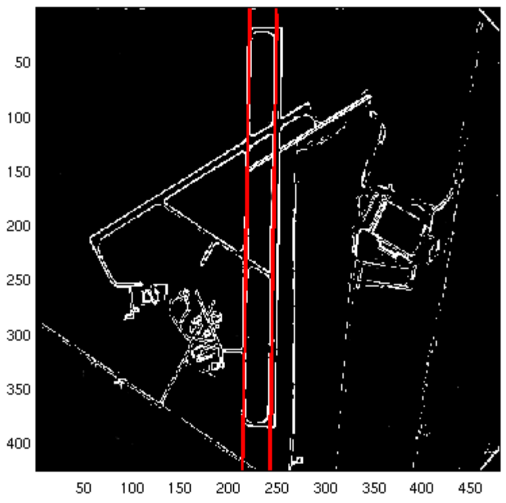

Figure 38: Reconstruction of detected lines

As we can see, with a very narrow confidence interval and a detection based on a small tolerance of values to be local maximum we can reconstruct the two main lines of the image which are the longest lines. If we reduce our tolerance value for maximum peak detection, due to the structure of Hough space we detect the same lines again. So we tried to increase the number of detected lines

by modifying the confidence interval for our angular value.
An important thing to notice here is that the main structure of the images are the two vertical lines that define the runway, this is why our area of interest is located around 180°. But other structures appears due to the various possible orientations due to the other components of the image.

If we consider the shape image the result we get is more surprising:

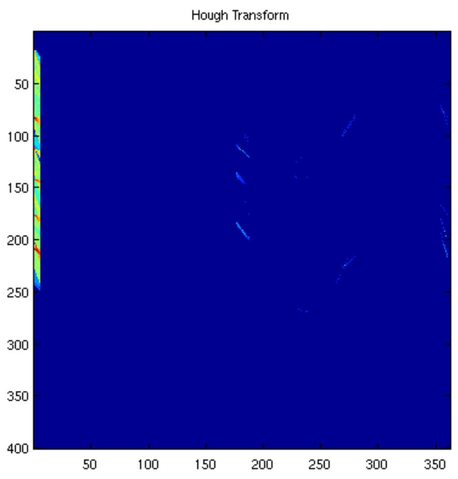

Figure 39: Log Hough parametric space using edge orientation

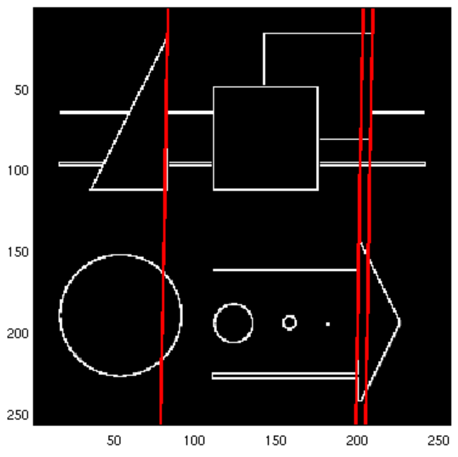

Figure 40: Reconstruction of detected lines

In the previous results, the log Hough space is only composed of few angular bands centred mainly around the vertical and horizontal orientations. It might appear weird and may suggest that this is a wrong result but the image is noise free so the gradient is defined everywhere with no variations. So when we apply our maximum detection technique we obtain the major vertical lines.

We are going to apply this method to a last image. This image is a binary version of a picture I took from down town SLC.

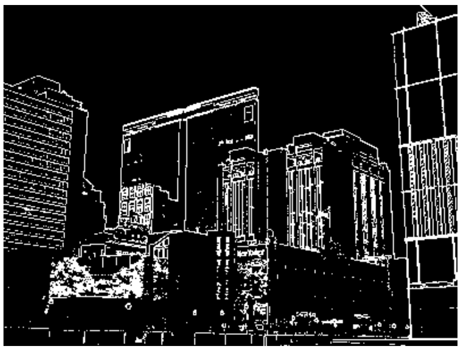

Figure 41: Binary input image

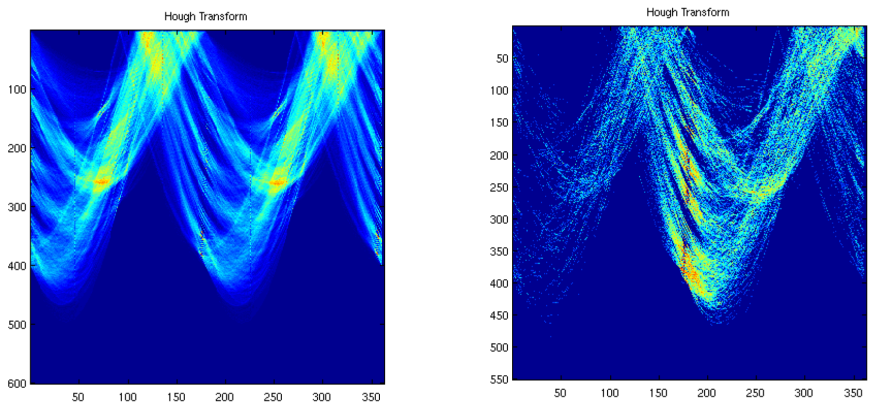

Figure 42: Smoothed Hough space and Hough space with reduced angular set

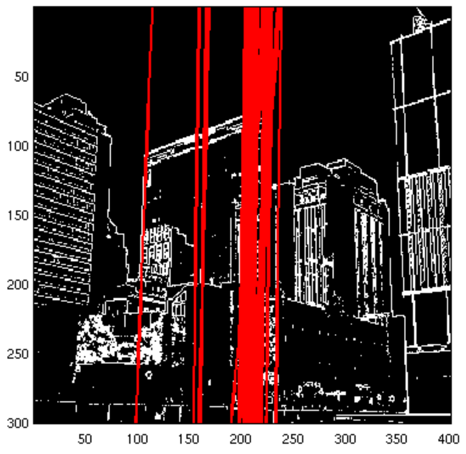

Figure 43: Reconstruction of detected lines

This image is mainly composed of vertical lines from the building. As we can see on the comparison between the two Hough space images, when we use the edge orientation we can reduce significantly the computation time.

## 3.4 Method Comparison

If we compare the two method we can notice a significant computation time reduction when using the edge detection. Indeed, we do not compute any more the parameters for the whole $\theta$ range but only for a reduced subset. The images we showed before illustrate the difference between the two computations because the resulting Hough Space is less cluttered so it explains that it takes less time to compute.

In terms of detection and line reconstruction using edge orientation is more accurate because it allows to compute only angular subsets of interest. So time computation increase significantly with image size but mostly with the number of edges and non background pixels.

We did have the ability to deeply studied the computational difference with accurate figures but the averaging difference is that applying Hough Transform using edge orientation is about 4 times faster than the other method. But this is really hard to express that because it depends of course on the size of the image, the number of pixels resulting from thresholding. So it would require more investigation to be able to quantify this aspect.

If we compare all the previous results we showed, we can see that they are pretty similar, the only thing to differentiate them is the number of lines detected bases on the technique used, because all them succeed in detecting the main edges.

## 3.5 Selective edge detection

This section is just an idea I got based on what we did before. If we know before applying Hough transform on an image the edge orientation, why not applying a selective edge detection based on statistical distribution of theses orientation. Indeed, the idea would consist in computing the gray level histogram of the edge orientation map and then center our Hough transform on a interval of values according to the gray level frequency. Here is an example.

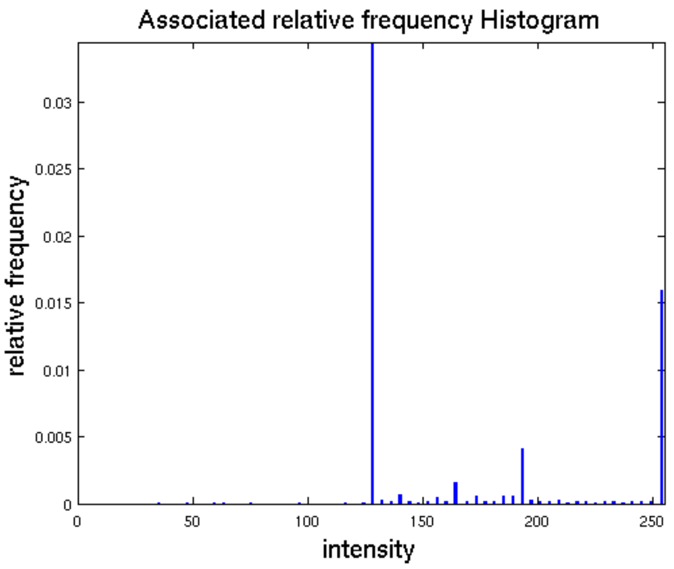

Figure 44: Histogram of the shape image

Here is the histogram of the geometric shape image. We know that the angle angular interval $[-180°, 180°]$ is spread over $[0, 255]$. So based on that we can locate clusters of angles based on frequency. There is a very important set of lines with a mid gray level which correspond to an angle of $0°$ which represent all the vertical lines. Then, we have an other cluster with intensity value of 255 which represent angles with a $180°$ angle so that's again vertical lines and that's what is in the structure of the image with mostly vertical lines. We can find a cluster at 193 which represents lines with an horizontal orientation. So based on this statistic description of the image structure we can tell what is the main orientation of lines and compute them according their orientation creating cluster of lines.

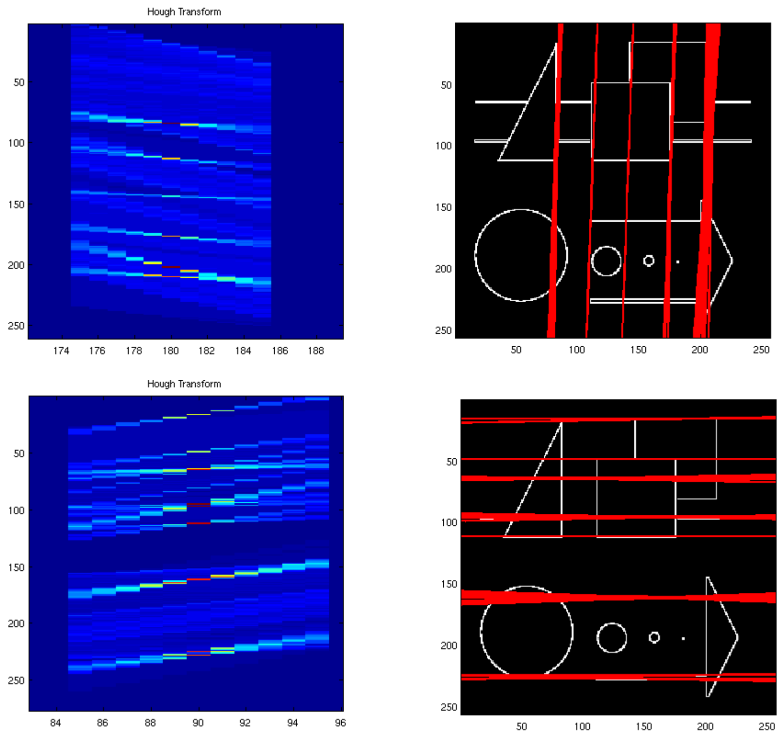

Figure 45: Extraction of vertical and horizontal clusters

On the previous picture we can clearly see all the characteristic crossing points aligned corresponding to parallel lines. In fact what we presented before due to line length is at stake here, because to have most of them we have to reduce the threshold and then we get more non local maxima. Using the decremental technique might help. the presence of other gray level is due to the presence of the circle that creates lines of all possible orientation because it has a theoretical infinite number of tangent lines. and but we can still try to extract some diagonal lines using the same approach. This part was just presenting a strategy that could be useful if we have the edge orientation image and if we want to extract lines based on their frequency of appearance in the image.

# 4 Implementation

Here is a presentation of the techniques we used to realise this project, the function and method we implemented to produce the images presented in the previous sections of this document.

## 4.1 The vote or score strategy

The vote strategy is an implementation technique we used to realize Hough transform. Indeed, we showed that for each point of the image space we map it into a sinusoid into Hough parametric space. We also said that all the points belonging to a common edge will have their sinusoid curve intersecting at a same location. So in order to be able to determine this location, we decide to use Hough space as an accumulator. Indeed, for each pixel where a sinusoid goes through we increment the value of this pixel. Then to determine the crossing points we will process this accumulator image we created.

```
for i=1:1:size(I2,1)
    for j=1:1:size(I2,2)
        if I2(i,j)==255
            Y(k)=i;
            X(k)=j;
            for theta=1:1:360
                rho = (X(k))*cosd(theta) + (Y(k))*sind(theta);
                if rho <=orig(1)
                    rho=orig(1)-rho+1;
                end
                 if rho > size(newI,1)
                    rho=size(newI,1);
                 end

            newI(round(rho),round(theta)) = newI(round(rho),round(theta)) + 1;

            end
            k=k+1;
        end

    end
end
```

The accumulator used for Hough space is a kind of 3 dimensional histogram so we can illustrate it with the following image:

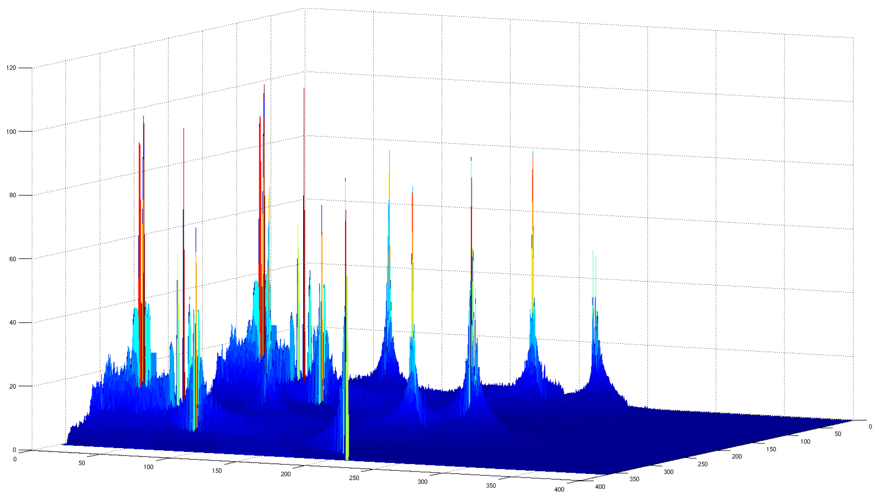

Figure 46: 3D view of Hough space

As we can see here, we have to detect exactly one pixel with the highest value per area, so we can clearly see on the previous picture that thresholding is not going to be the best method to do so.

## 4.2 Thresholding method

To perform this technique we just need to define a value as threshold. To illustrate, we can say that it's similar to define, in the previous image, a plane with this value as height. Everything above will be considered as interesting, everything below will be ignored.

```
k=1;
for i=1:1:size(newI,1)
    for j=1:1:size(newI,2)
        if (newI(i,j)) >= threshold
            maxi_rho(k)=i;
            maxi_theta(k)=j;
            k=k+1;
        end
    end
end
```

Note that the variables *maxi_rho* and *maxi_theta* are storing the coordinates in Hough space as requested in question ($c$) to get the vote value we just need to read the value of the accumulator associated to those variables. We did not write files containing those values, the line reconstruction was used a sanity check to see if our detection was working.

### 4.3 Filtering method

Another technique which work pretty well is to use a Gaussian smoothing to reduce noise and value dispersion allowing us to have a clearer image with hopefully clearer maxima.

```
weight=[1/16,2/16,1/16;2/16,4/16,2/16;1/16,2/16,1/16]
 for i = ceil(size(weight,1)/2) :1: size(newI,1)-size(weight,1)+ceil(size(weight,1)/2)
    for j = ceil(size(weight,2)/2) :1: size(newI,2)-size(weight,2)+ceil(size(weight,2)/2)
        convol=0;
        %compute convolution for the neighbourhood associated to the kernel
        for a = 1:size(weight,1)
            for b=1:size(weight,2)

            convol = convol + (weight(a,b)*newI(i-a+ceil(size(weight,1)/2),j-b+ceil(size(weig

            end
        end
        newI(i,j)=convol;
    end
end
```

And as we can see on the following picture of Hough space we end up with something a bit better:

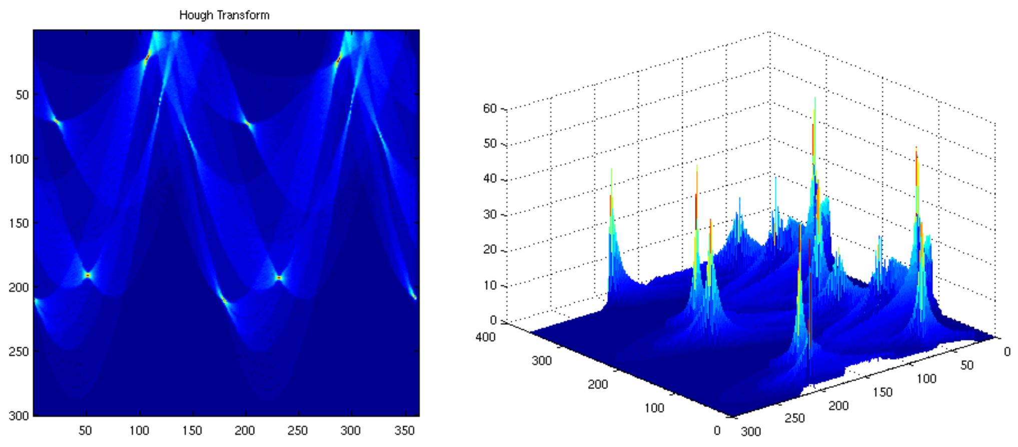

Figure 47: Regular and 3D view of smoothed Hough space

### 4.4 Decremental method

In the first pass of the image, when we compute Hough space, we stored pixels which belong to an edge. By doing so, we are able then to apply the second pass only based on those pixel which will save some computational resources. Indeed, this worked because we where working with binary images where knowing if a pixel belongs or not to an edge is easy to determine. Then we re compute the Hough sinusoid associated to those points and decrement their value in the accumulator if they are below a certain value. It may not be the best implementation of this technique but it seems to be working.

```
 for i=1:1:size(X,2)

        for theta=1:1:360
            rho = X(i)*cosd(theta) + Y(i)*sind(theta);
                if rho <=orig(1)
                    rho=orig(1)-rho+1;
                end
                 if rho > size(newI,1)
                    rho=size(newI,1);
                 end

            if  (newI(ceil(rho),round(theta))<=maximum)
                newI(ceil(rho),round(theta)) = newI(round(rho),round(theta)) - 1;
                if newI(ceil(rho),round(theta)) <= 0
                    newI(ceil(rho),round(theta)) = 0;
                end
            end
        end
 end
```

## 4.5 Using edge orientation

Here, we need to use another input image providing us with the local edge orientation. This aspect has already been implemented and studied in project 2. It requires to compute the local gradient of the image and then we can derive a magnitude map and an edge orientation map. We will use the images produced during this project as input for our function. This function will associate this angular value to the computation of Hough transform to reduce it to a small subset of angles associated to each point. By doing so, we can reduce a lot the clutter of the Hough parametric space and hope to be able to find local maxima easily. The following code present the new computation of Hough space based on this angular information and also based on the confidence interval we provide due to the noisy structure of this image.

```
for i=1:1:size(I2,1)
    for j=1:1:size(I2,2)
        if I2(i,j)==255
            Y(k)=i;
            X(k)=j;
            theta_im(k) = I3(i,j)*(360/255);
            theta_lower = round(theta_im(k)) - 5;
            if (theta_lower <=0) theta_lower = 1; end
            theta_upper = round(theta_im(k)) + 5;
            if (theta_upper >360) theta_upper = 360; end
            for theta=theta_lower:1:theta_upper
                rho = (X(k))*cosd(theta) + (Y(k))*sind(theta);
                if rho <=orig(1)
                    rho=orig(1)-rho+1;
                end
                 if rho > size(newI,1)
                    rho=size(newI,1);
                 end

            newI(round(rho),round(theta)) = newI(round(rho),round(theta)) + 1;

            end
            k=k+1;
        end

    end
end
```

## 4.6 Other functions

We used few other functions we implemented in previous projects to study histograms and compute edges of an image. Sometimes a small external processing was done to enhance the quality of the input image and obtain a bit more accurate results.

## 5 Conclusion

This project allowed us to study again an important feature in image processing which is edge detection. We already studied how to compute them using filtering. In this project we implemented Hough transform which rely on a dual space called Hough space. This dual space is using accumulation techniques to convert an edge into a set of curves and determine the edge parameters : slope and y-intercept.

We could see in that project the importance of a pre processing step in order to provide a better image to our algorithm. Indeed, an edge detection is performed here and then a thresholding of the image to only extract features of interest as clear edges. We also saw that the structure of these edges will influence the Hough space structure and will have significant consequences on multi line extraction.

I also show in this project that using mathematical functions according to their behaviour was a good solution to improve and enhance the contrast of the Hough space without having to use another editing software. This is just a global intensity processing function we studied at the beginning of the semester. Of course using other function is possible and could probably lead to better results.

As we've seen, this edge computation was done using an edge map extracted using the gradient of the image or other types of filter. Other method exists, one of them rely on mathematical morphology to process images. The goal would be to extract the topological skeleton which is a line version of a shape which is equidistant to its boundaries.

We will show in the next project that there are other method based on active contours to detect the borders of an object in an image. This method also relies on specific features of the image as we did in this project. The final goal is always to be able to segment object from an image to study them and their properties.

## References

[1] Wikipedia contributors. Wikipedia, The Free Encyclopaedia, 2012. Available at: http://en.wikipedia.org, Accessed October-November, 2012.

[2] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Third Edition, Pearson Prentice Hall, 2008.

[3] G. Gerig. Linking Image-space and accumulator space: a new approach for object-recognition. ICCV87, p 112-117.

[4] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes.

[5] P.V.C. Hough. Method and means for recognizing complex patterns, u.s. patent 3069654. 1962.

## List of Figures