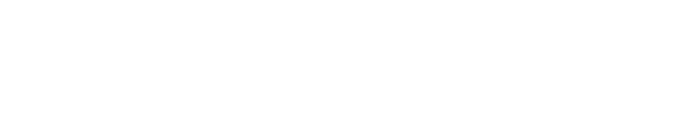
Final Project

Image Processing

College of Management Academic Studies

Computer Science





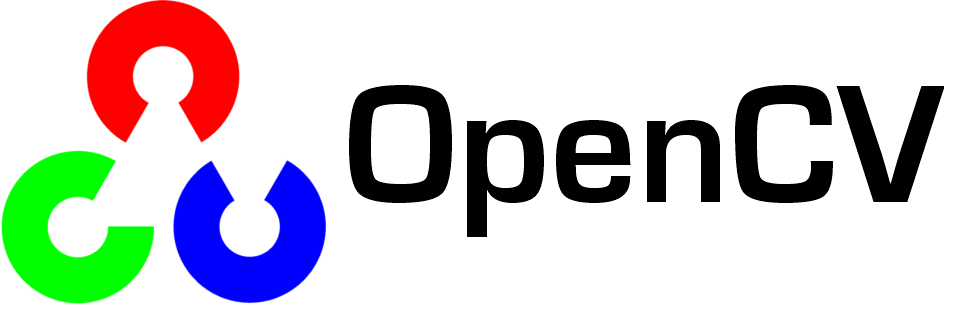


Table of Contents

[Submission Details 2](#_Toc480388197)

[Part A: Single image processing 3](#_Toc480388198)

[Question 1: Histogram Equalization Algorithm 3](#_Toc480388199)

[Print screen from MATLAB 3](#_Toc480388200)

[Question 2: Image sharpening 4](#_Toc480388201)

[Print screen from MATLAB 4](#_Toc480388202)

[Question3: Image compression using FFT 5](#_Toc480388203)

[Print screen from MATLAB 5](#_Toc480388204)

[Question 4: Finding connected components in a binary image 6](#_Toc480388205)

[Print screen from MATLAB 6](#_Toc480388206)

[Part B: Completion of straight lines using Hough Transform algorithm 7](#_Toc480388207)

[Question 1 (15 Points) 7](#_Toc480388208)

[Part C: Image watermarking 7](#_Toc480388209)

# Submission Details

**Submitted By:**

Nimrod Dickler – 301860698

David Akler – 203925524

Moran Shmulovich – 201559192

**Submitted Due date:**

28.03.2017

**Submitted To:**

Dr. Moshe Butman

# Part A: Single image processing

The answers in the section are written in MATLAB.

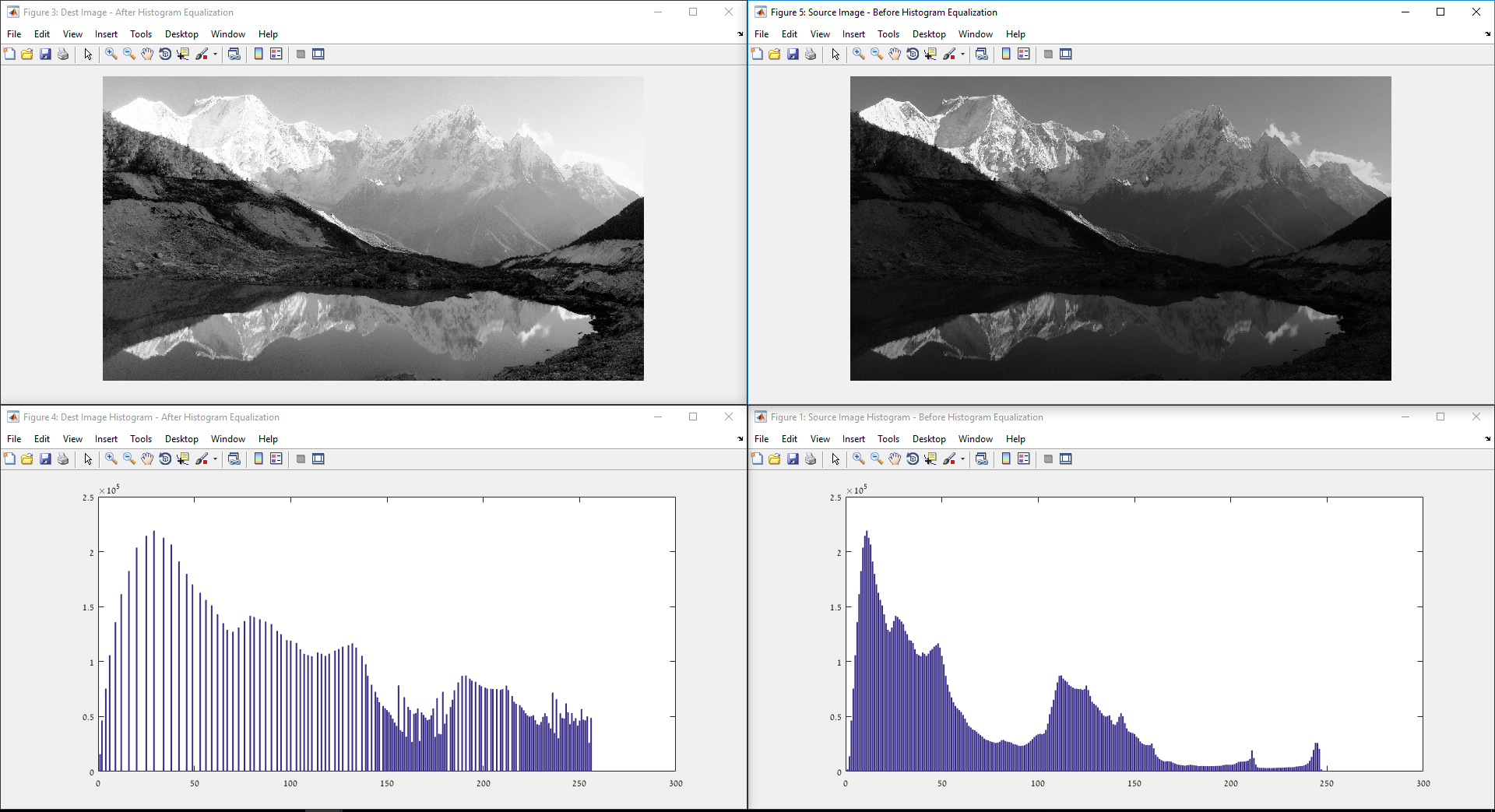
## Question 1: Histogram Equalization Algorithm

In this question we were asked to create a histogram equalization function,

We created a function that calculates the frequency of each grey level and then the probability for each gray level in our image (between 0-1) and the cumulative probability.

By applying the cumulative probability on to our original image we will eventually get a new image with a much more balanced intensity of grey and a little impact to the image as possible.

### Print screen from MATLAB



## Question 2: Image sharpening

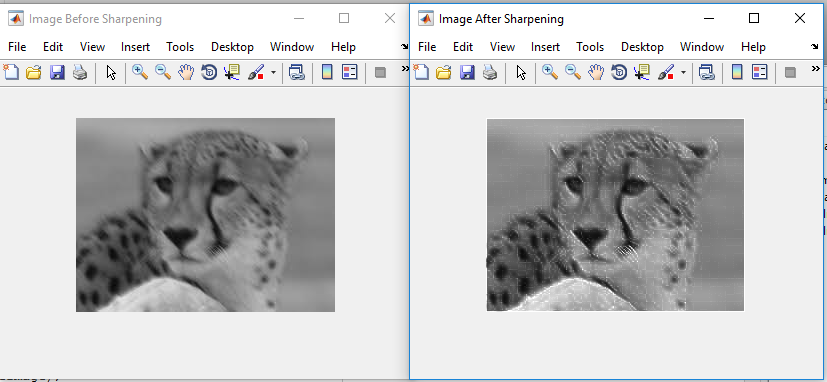
In this question we were asked to sharpen an image,

……Not Done….

We used a 2D convolution with a kernel of 3x3 with 0.111 to smooth the image

Than we used reduced it from the image and multiplied by the sharp rate

### Print screen from MATLAB



## Question3: Image compression using FFT

In this question we were asked to sharpen an image,

……Not Done….

We used a 2D convolution with a kernel of 3x3 with 0.111 to smooth the image

Than we used reduced it from the image and multiplied by the sharp rate

### Print screen from MATLAB

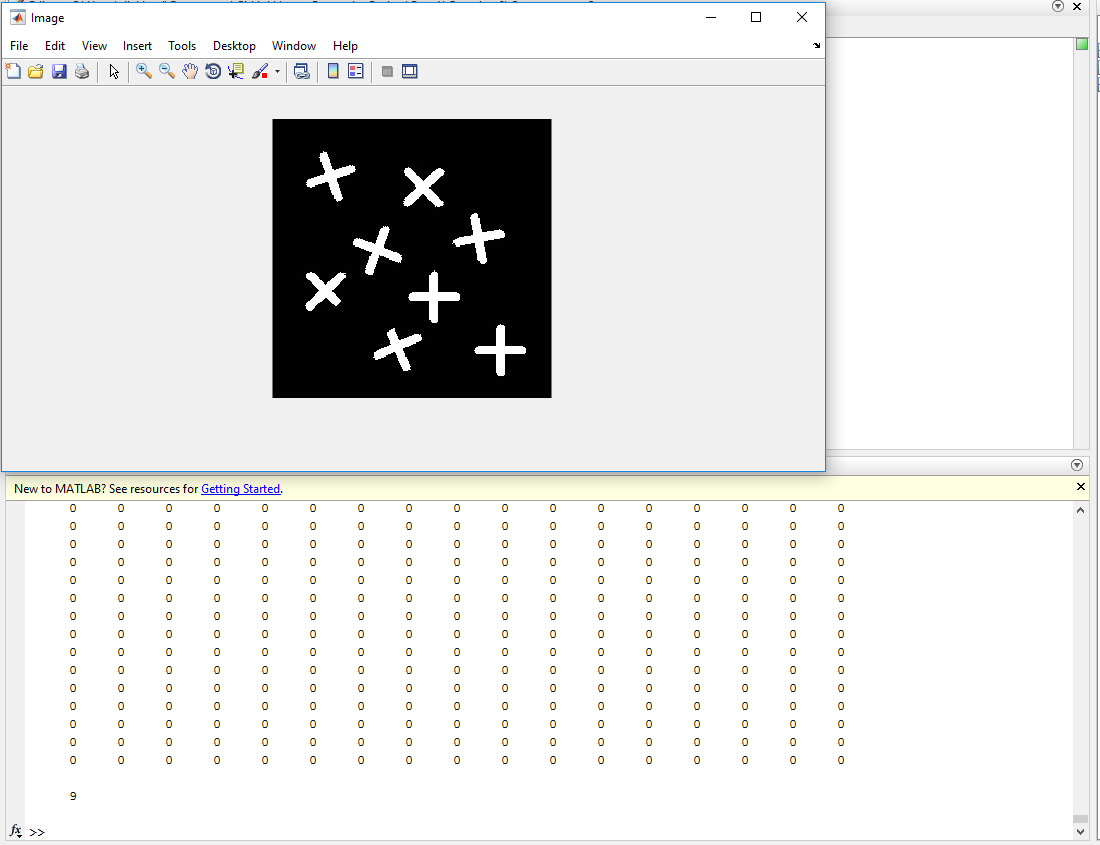
## Question 4: Finding connected components in a binary image

In this question we were asked find the number of connected components in a binary image,

We implemented a function that counts the connected components according to a threshold.

We traverse our image and check if we have a white pixel that wasn’t visited and then check all his neighbors (from all angles) to see if they are also white, meaning the same component, all white neighbors we found will be added to a stack we will do the same checks on them, when we don’t have any white neighbors in the stack that means we found all the whites in the component, so we raise the component counter, next component will be identified as 2 in the visited matrix.

### Print screen from MATLAB



# Part B: Completion of straight lines using Hough Transform algorithm

The answers in the section are written in Python using the open-cv library.

## Hough Transform Lines Detection

In this question, work on the implementation Hough transform line detection,

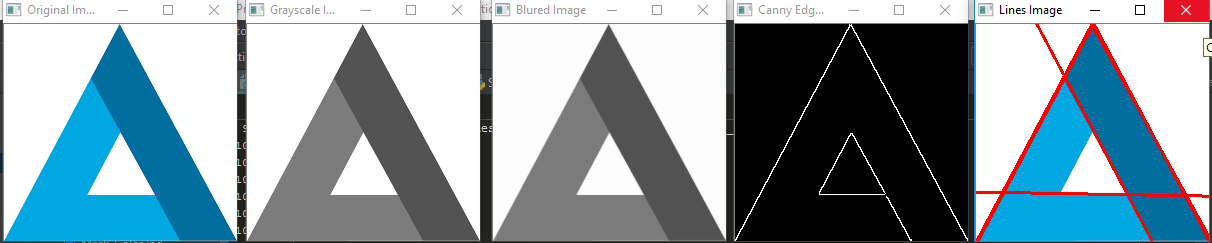
……Not Done….Our function (hough\_transform\_line\_detection) will get a grey scale image, run Gaussian blur to filer noise from the image, then run the canny edge detection algorithm from open-cv on the image to get the edges of the image.

On our edge image we run the Hough transform algorithm, the algorithm

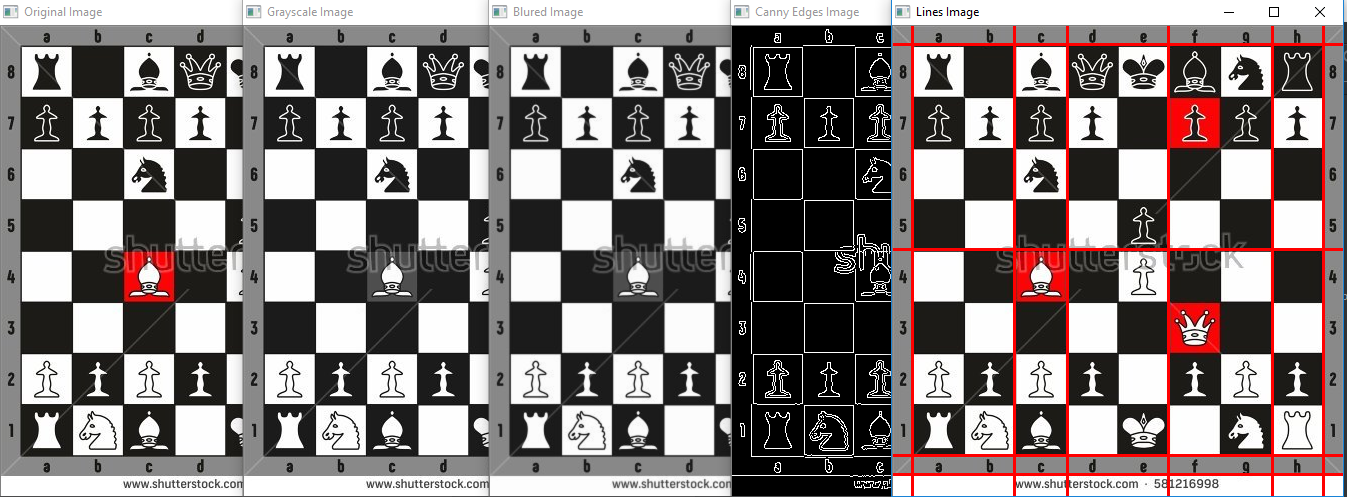
We create an accumulator – an array in the Hough space that that has all the edges and

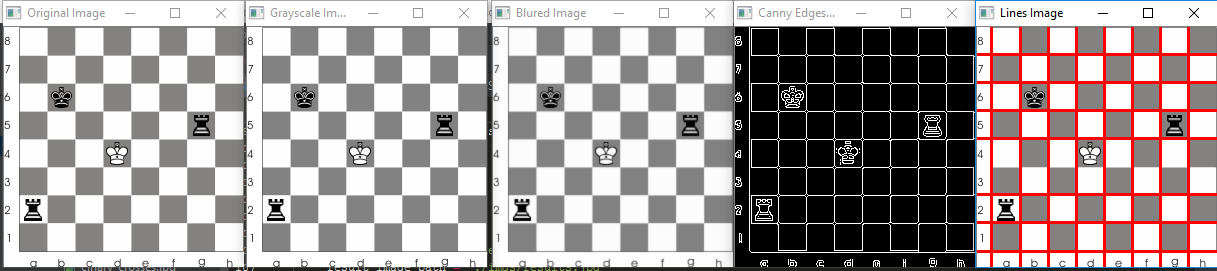
Than we find N peaks in the accumulator and write them on the original image.

### Examples for our Hough Transform Implementation



# 





# Part C: Image watermarking

The answers in the section are written in MATLAB.

## Image Watermarking

In this question, we decided to implement image watermarking,

We chose an image and implemented a rectangle we created manually inside the image.

In the process, we worked on each color channel of the image (Red, Green and Blue)

We converted each color channel using discrete cosine transform and we did the same thing to our rectangle and then we joined them together and did an invert discrete cosine transform.

After that we show that we can’t see the rectangle we hid inside the image but when checking the real image compared to the original image we can see the differences.

After that we did showed how we extract the rectangle from the image and get the original image again.

Using image watermarking we can secure our images a little more and validate authenticity of an image, and all that happens almost without impact to the image (or the end users experience, because the image did change but the user couldn’t feel that).

### Print screen from MATLAB

……Not Done….