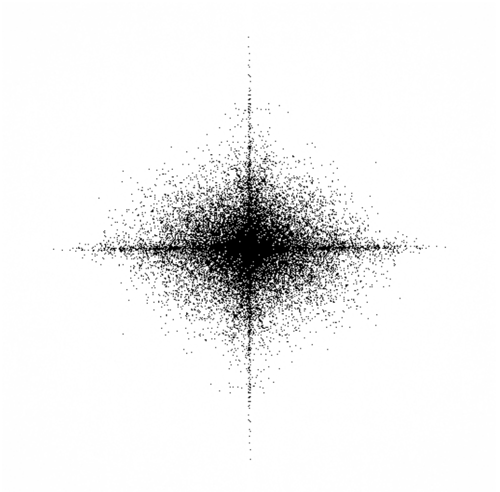
CSCI B 657 Computer Vision
Assignment - 1

Part 1.1

We are plotting the fourier transform of the imagePNGInputImage512.png



We observed that using log10 gave visually better results than using loge.
Please refer “fft_magnitude” for code implementation.

Part 1.2

This function is used to remove the interference. The real part of fft of noise1.png contained “HI” as the noise. The log of magnitude of this image had components whose values was less than 0.12 (excluding the 256th row in the fft matrix).All those values were made to zero which removed the noise/ interference in the image.
Now, the resultant image after noise removal was not as bright and required enhancement. Hence using trial and error method we came up with 2 coefficients to boost the frequency values in fft matrix.
4.60625 was multiplied with all the values in the real matrix and imaginary matrix.
The Dc term was multiplied with 1.47078125.
We also observed that after multiplying the above values the resultant image had frequencies in the same harmonic resulting in a image without distortion and had better brightness.
Please refer “remove_interference” for code implementation.

Part1.3

There are 2 parts to this problem.

1) Add watermark

We generated a random number using srand() that generates a constant random number for a particular given input N.
This number was converted to **binary of length l** where **2l is the number of bins** that had been modified to inject the watermark. The water mark bins were injected with a **space of 2 bins**. We handled both odd and even cases of l and calculated the radius of the circle accordingly. (Please refer the code "mark_image")
Watermark was added to the image using the equation given in the question. We found **alpha = 0.65** that helped in threshold setting for checking if watermark is present which is explained below.

2) Check if watermark exists

An input image is passed with a parameter N. So the random number is generated again in the similar way .
After extracting the values from the same circle, we observed that to declare if the watermark was present in an image, we needed to find out Pearson coefficient between the generated binary number and extracted values.
However, peasron coefficient can be calculated between continuous variables.
Hence we calculated 2-sample Welch's t-test. And using the absolute value of the resultant value, we are deciding if the watermark is present or not.

The threshold value is varying between 0.0 to 1.8 if the watermark is not present.
The threshold value is varying between 1.1 to 4.7 if the watermark is present.
(These values are absolute values)

Depending on the coefficient calculated, the code prints if the watermark with the given input number is present or not.

"We need to find the correlation between Vi and R'(x, y)
As we know that Vi contains only 0 and 1's (Categorical or binary values), but R'(x, y) is a continuous value. In order to find the correlation between these two we use a different method other than using Pearson correlation coefficient, i.e., two sample test.
Two sample test:
We have values (0,1) and we got their respective values of R'(x, y). We are going to take these values into 2 samples, one contains all the values of R'(x, y) when Vi = 0 and the other contains all the values of R'(x, y) when Vi=1. We need to find whether these sample values are different from each other or not. If they are different then we can say that there is a correlation between R'(x, y) and Vi and vice-versa. So, we calculate the sample means, variance and number of values for each sample. We take the difference of means of these 2 samples (meand) and calculate the effective variance (evar) of these 2 samples by Welch's t-test.
Null hypothesis: meand = 0 (or) there is no difference between the 2 samples (or) no correlation between Vi and R'(x, y)
Alternative Hypothesis: meand != 0 (or) there is difference between the 2 samples (or) correlation between Vi and R'(x, y)

We assume that although this forms a t distribution with some degrees of freedom, we consider this as a normal distribution
Z = meand/evar
We need to keep a threshold for the Z value inorder to accepting or rejecting the null hypothesis.

"

Part 2: Detecting Objects

What we have done?

Initially, the convolution functions convolve_general and convolve_seperable has been written to convolve the image matrix.

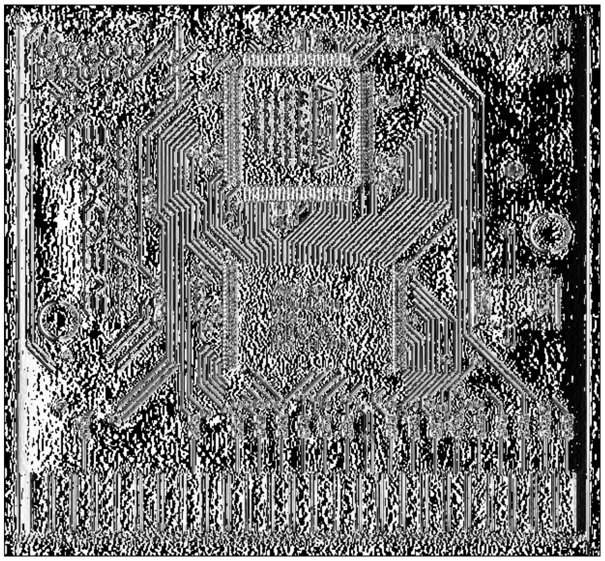
Functions:
Convolve_general: It takes the input image and a 3X3 mean filter.

Convolve_seperable: It takes the input image and a row Gaussian filter and a column Gaussian filter.

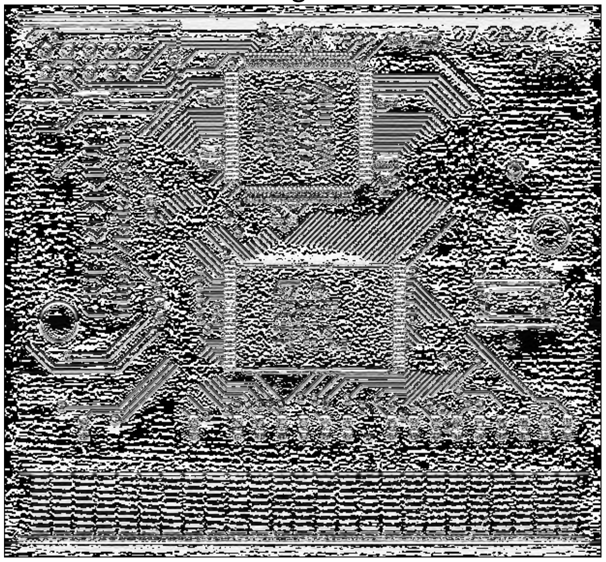
filter_creation: It takes the sigma value and creates the filter based on the sigma value. (Ref: Digital Image Processing Fundamentals Page: 115)

sobel_gradient_filter: [1 2 1] [-1 0 1]^T gradient filter is applied to the image the get the X derivative, and [-1 0 1][1 2 1]]^T is applied to the image to get the Y derivative.

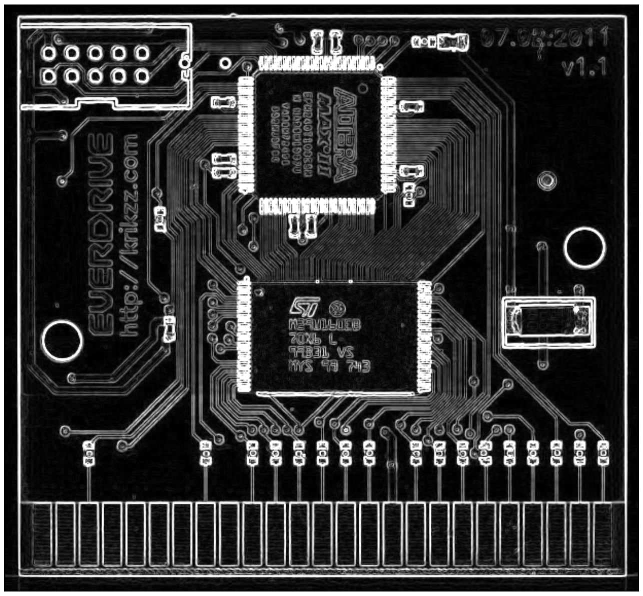
Sobel X gradient



Sobel Y gradient



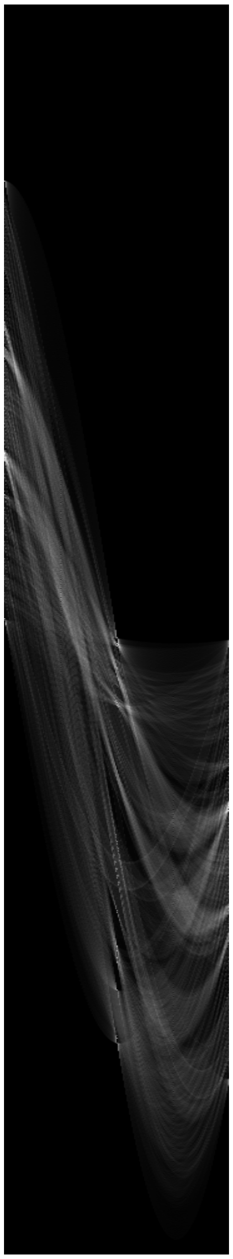
gradientMagnitude: This function takes the X gradient and Y gradient, computes the gradient and return the matrix.

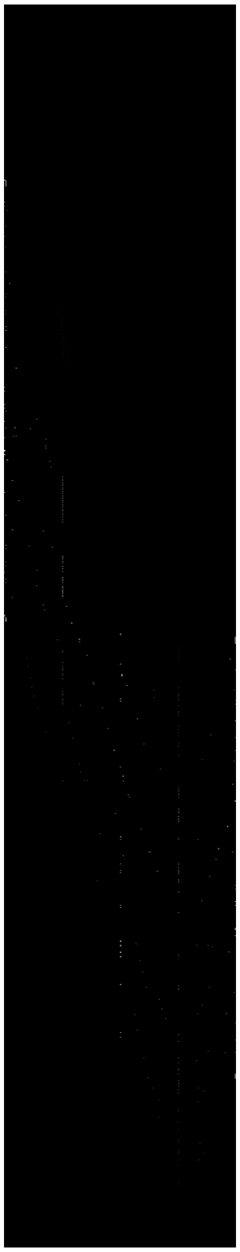


find_egdes: find_edges take the input image, and generates the gradients in X and Y direction. Gradient Magnitude is calculated by taking the sqrt of sum of squares of the X gradient and Y gradient.

Approach 1: Hough transform
For this first the image is smoothened with the Gaussian filter and the sobel operator in X direction and Y direction is calculated.
Gradient magnitude is changed to binary image with 255 if the value is greater than 150 and 0 otherwise.

Now the edge points are converted to polar coordinates Below is the plot of the accumulator for the points in binary image.


Hough Accumulator

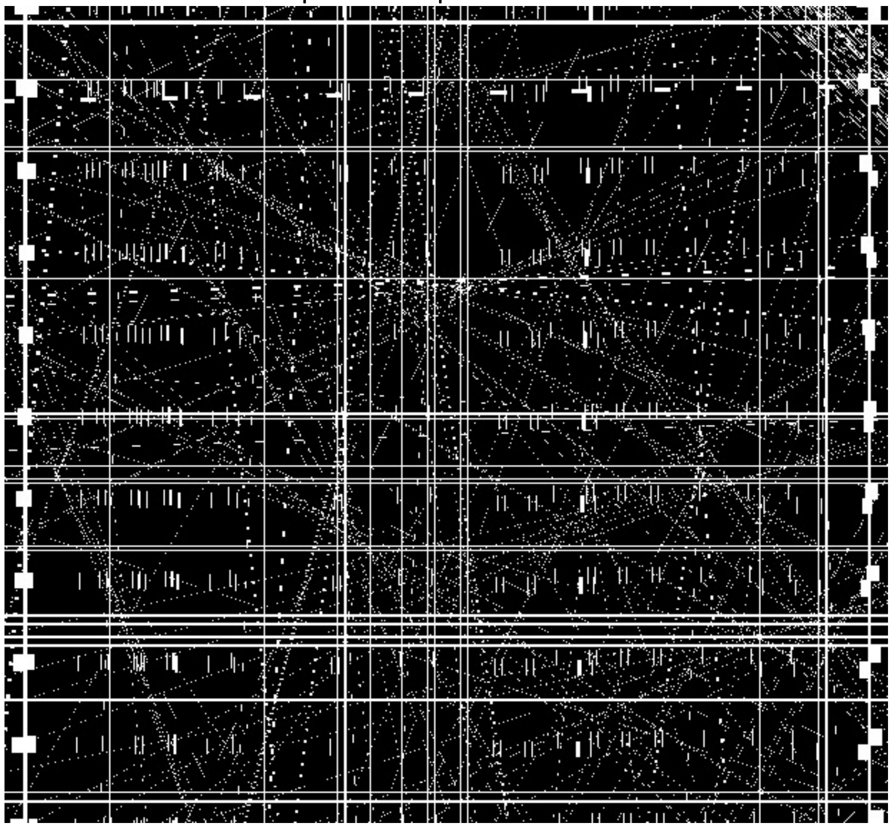

Accumulator Peaks

Here the X axis ranges from -90 to 90 and Y axis ranges from –rMax to rMax(max distance from center to image end point)

Non-maximal Suppression:
Here to find the peaks we have implemented the non-maximal suppression with the bin size of 2X2 and picking the local maxima in it.

Here we had a problem to detect the relevant edges from the lines

This is the output of the peaks of the accumulator.



Approach 2: Sliding Window:

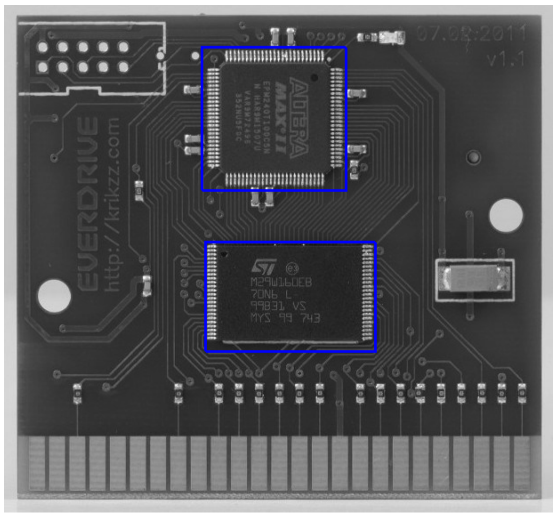
Since we weren't successful to determine interest points from Hough transform, we have implemented sliding window approach.

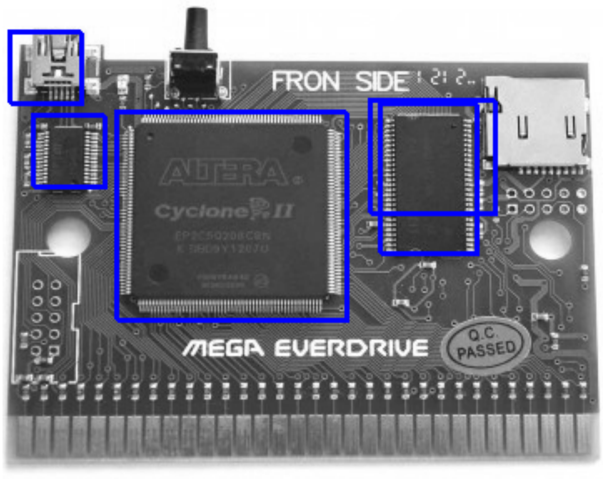
Here, we have used car templates to compare with the original image. A cross correlation is applied to the template and the image part under consideration to find the similarity, here a high value is cross correlation gives us the peak in similarity and return the position of the ICs.

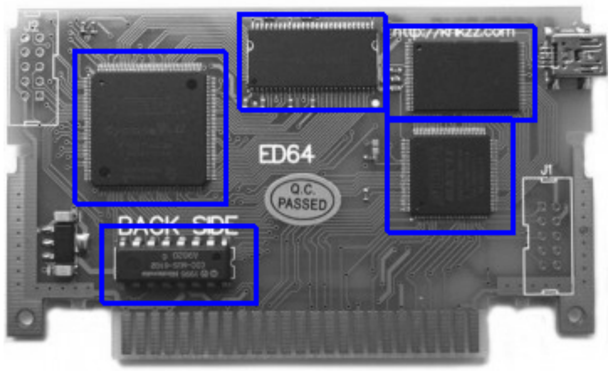
Here the image, the sobel operator was applied in X and Y direction, and the difference in gradient is calculated.
When the template resolution is similar to the image, we don't compare the image since there will be a less probability of having a large IC.

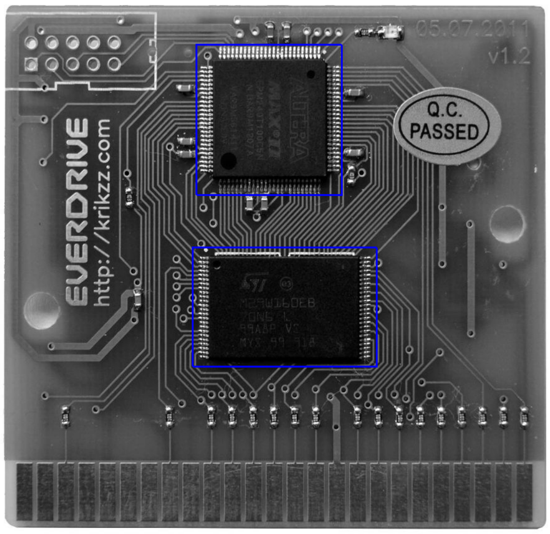
The problem with the sliding window is that the resolution of the image and the templates, we could see noise in the output image.

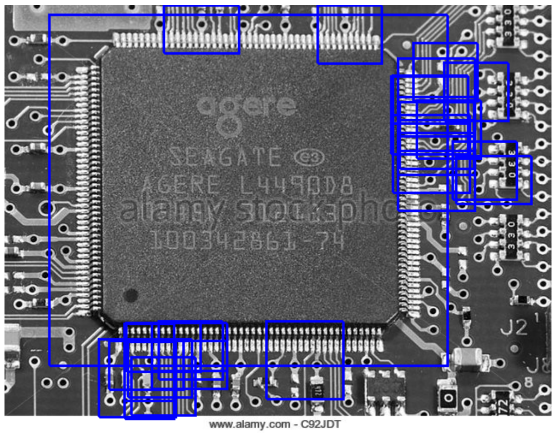
Here are results of this approach:


07.08.2011
v1.1
ALTERA MAX II
M29W160EB
70N6 L
99B31 VS
MYS 99 743
EVERDRIVE
http://krikzz.com


FRON SIDE
ALTERA
Cyclone II
MEGA EVERDRIVE
Q.C. PASSED


http://krikzz.com
ED64
Q.C. PASSED
BACK SIDE
J1


05.07.2011
v1.2
Q.C. PASSED
ALTERA MAX II
EVERDRIVE
http://krikzz.com
M29W160EB


agere
SEAGATE
AGERE L449DD8
100342861-74
J2

Reference:

http://ieeexplore.ieee.org.proxyiub.uits.iu.edu/stamp/stamp.jsp?tp=&arnumber=5375779
http://www.cs.utoronto.ca/~fidler/slides/CSC420/lecture17.pdf
http://www.ics.uci.edu/~majumder/DIP/classes/EdgeDetect.pdf
http://me.umn.edu/courses/me5286/vision/Notes/2015/ME5286-Lecture9.pdf
https://pdfs.semanticscholar.org/37c2/eba4ab5959891c45912c8cd8b6b8f0734026.pdf