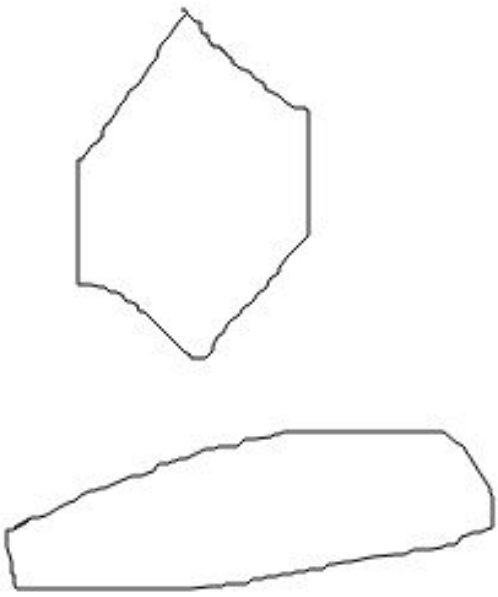
Assignment 1

Name: Sayak Kundu
Roll Number: 20161035

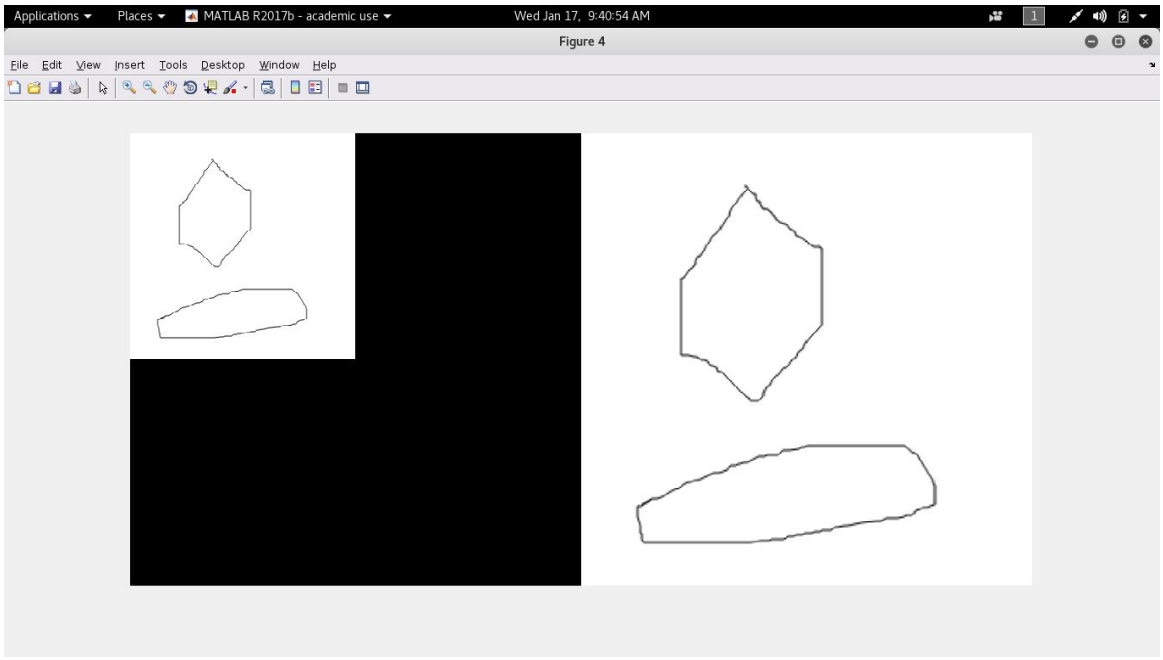
Question - 1
X = 2

1. Original

Ellipse and polygon



On expansion (Montage Mode)



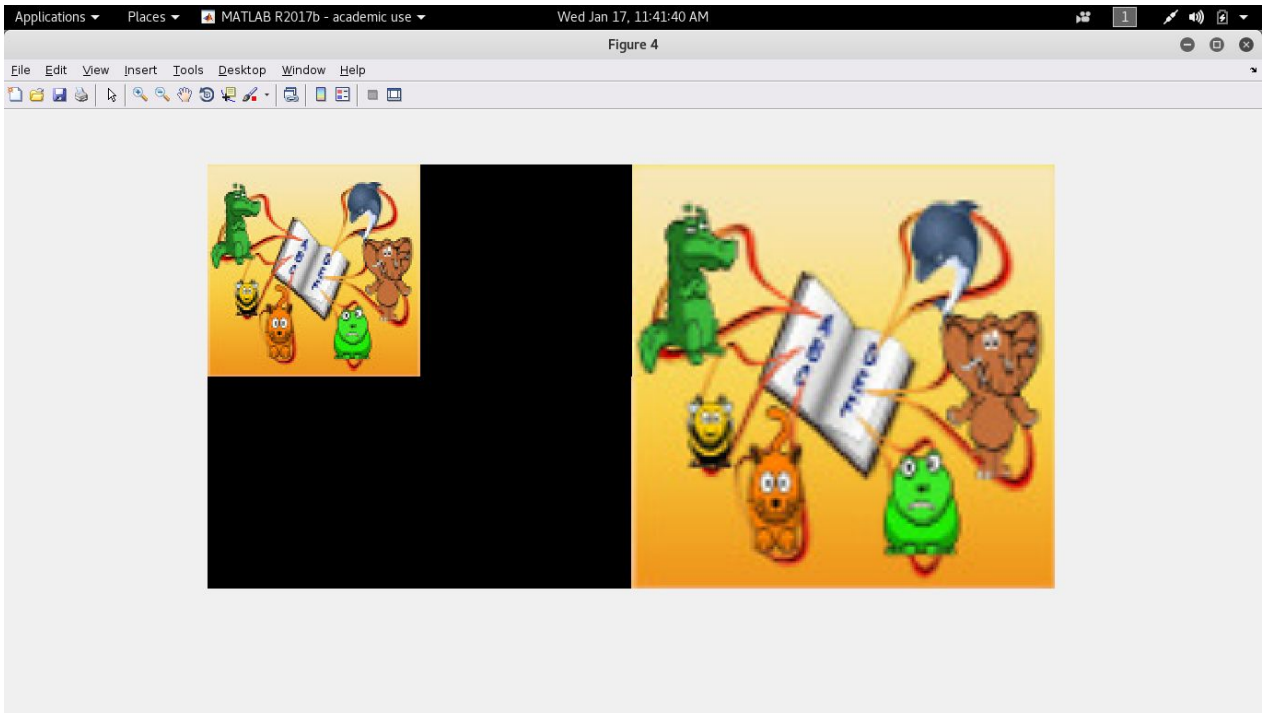
Note: Image is too big to fit on screen; displaying at 67%

2. Original (Black and White)

On expansion (Montage Mode)

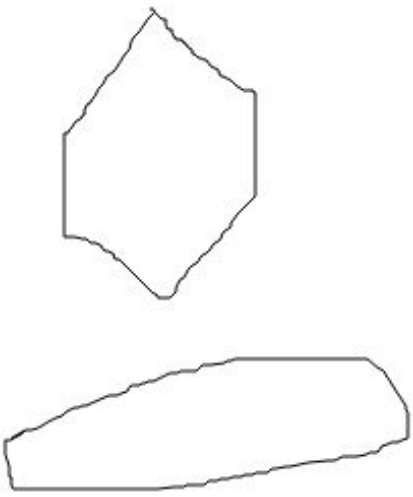
3. Original (Coloured)

On expansion (Montage Mode)

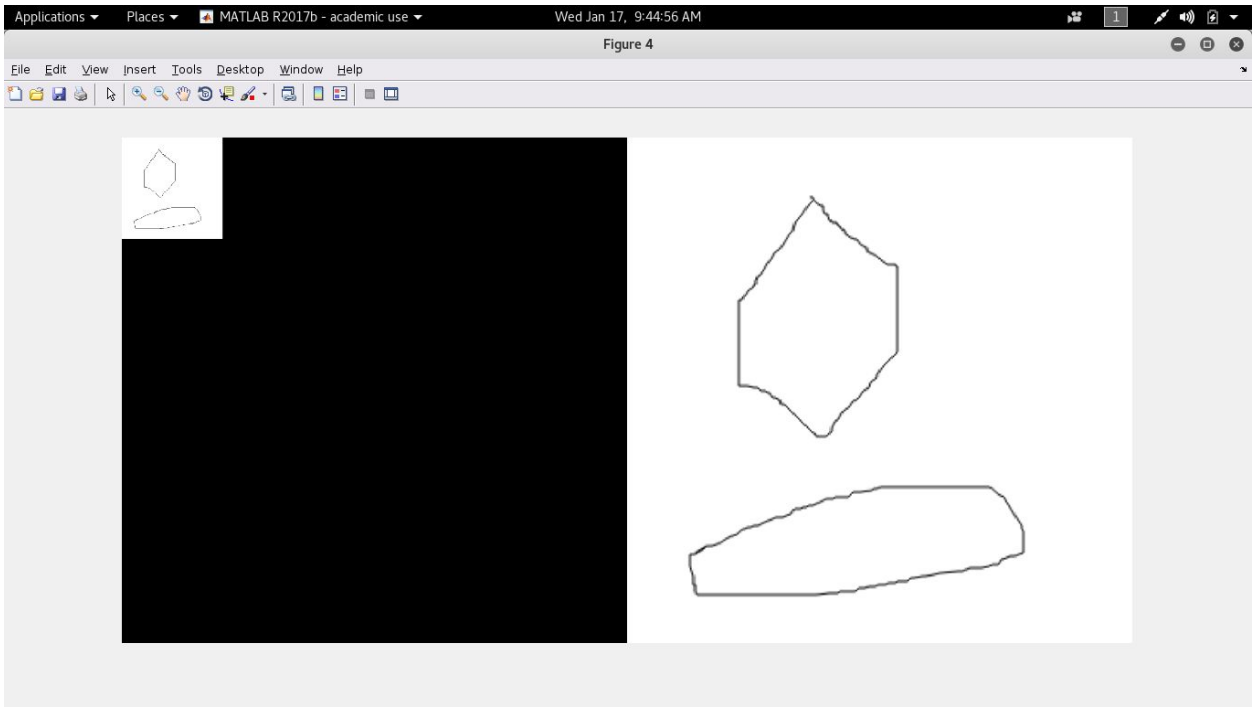


X = 5

1. Original



On expansion (Montage Mode)



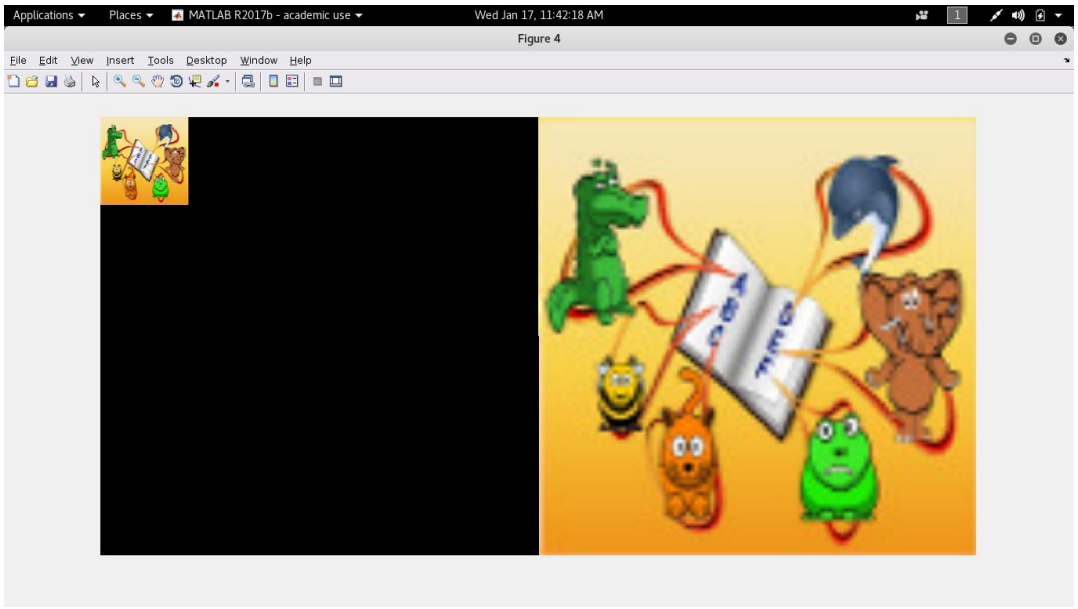
Note: Image is too big to fit on screen; displaying at 67%

2. Original (Black and white)

On expansion (Montage Mode)

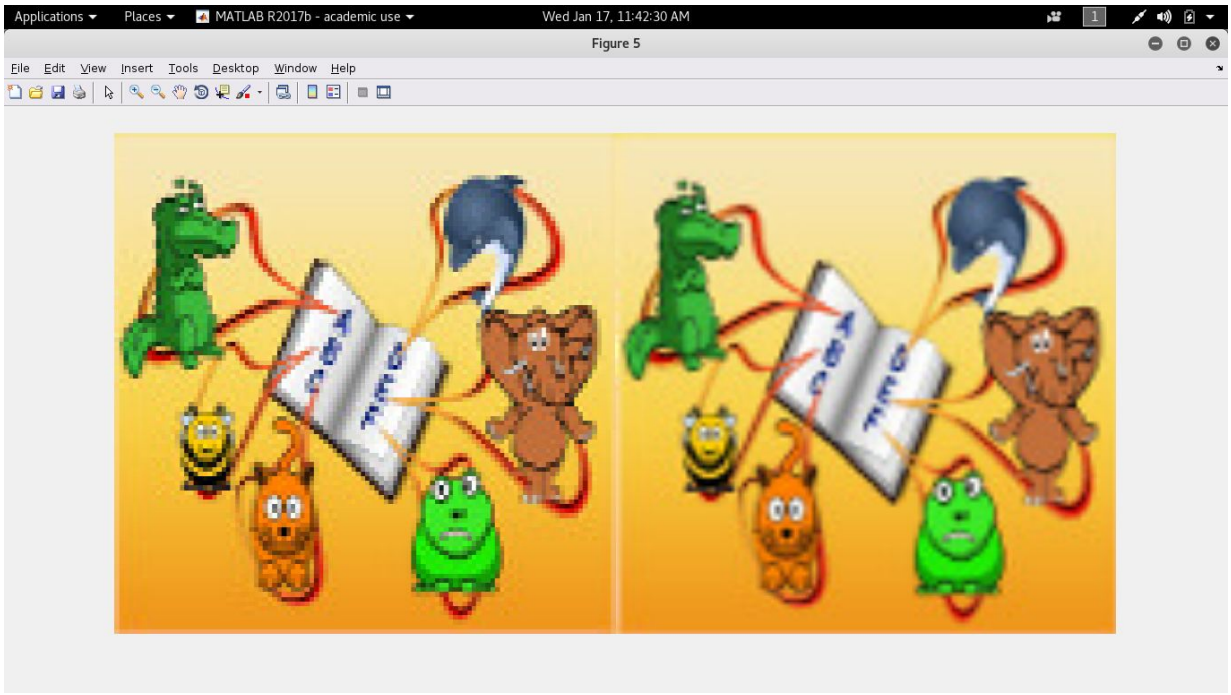
3. Original (Coloured)

On expansion (Montage Mode)



Difference between Nearest neighbour interpolation and Bilinear interpolation

Coloured



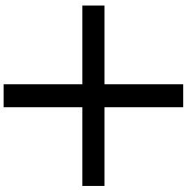
Black and White

We can clearly see that the left image is more pixelated that the others in both the cases. The left uses **Nearest Neighbour Interpolation** for resizing whereas the right one uses **Bilinear Interpolation**.

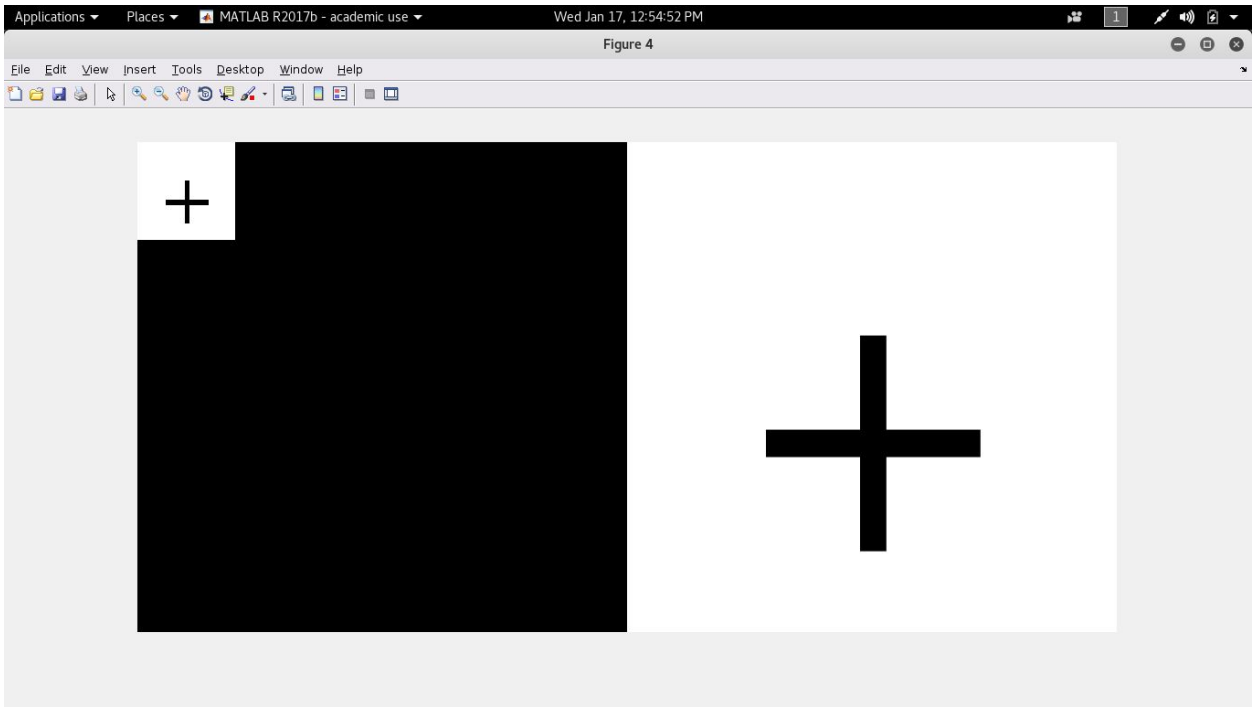
Nearest Neighbour fairs equally or even better with images containing high number of straight edges (horizontal and vertical).

For example:

Original

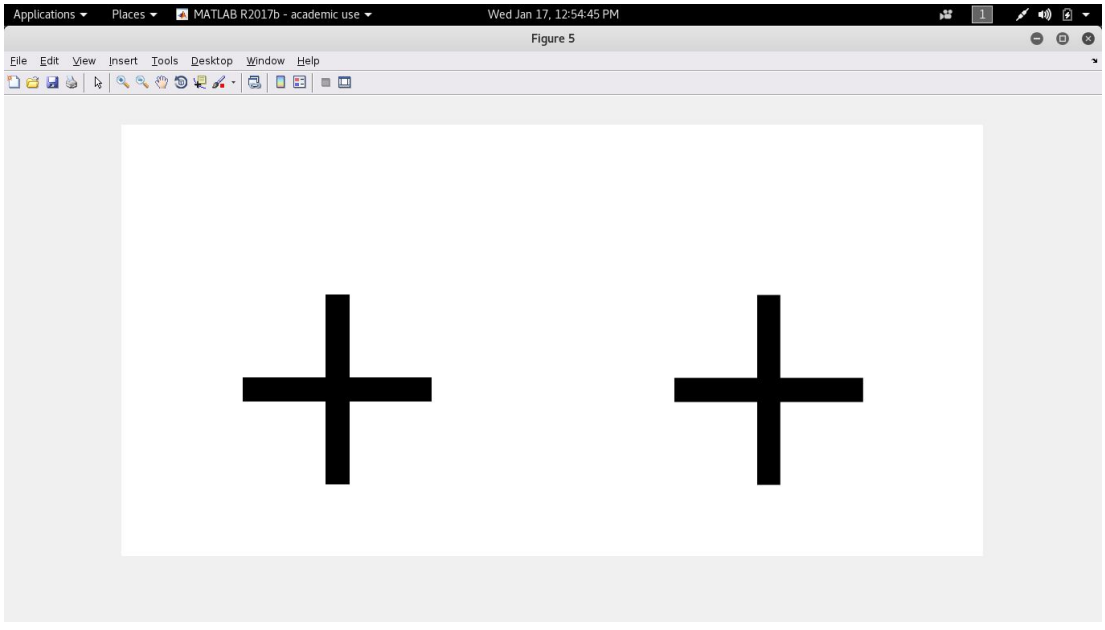


X = 5 (Montage Mode)



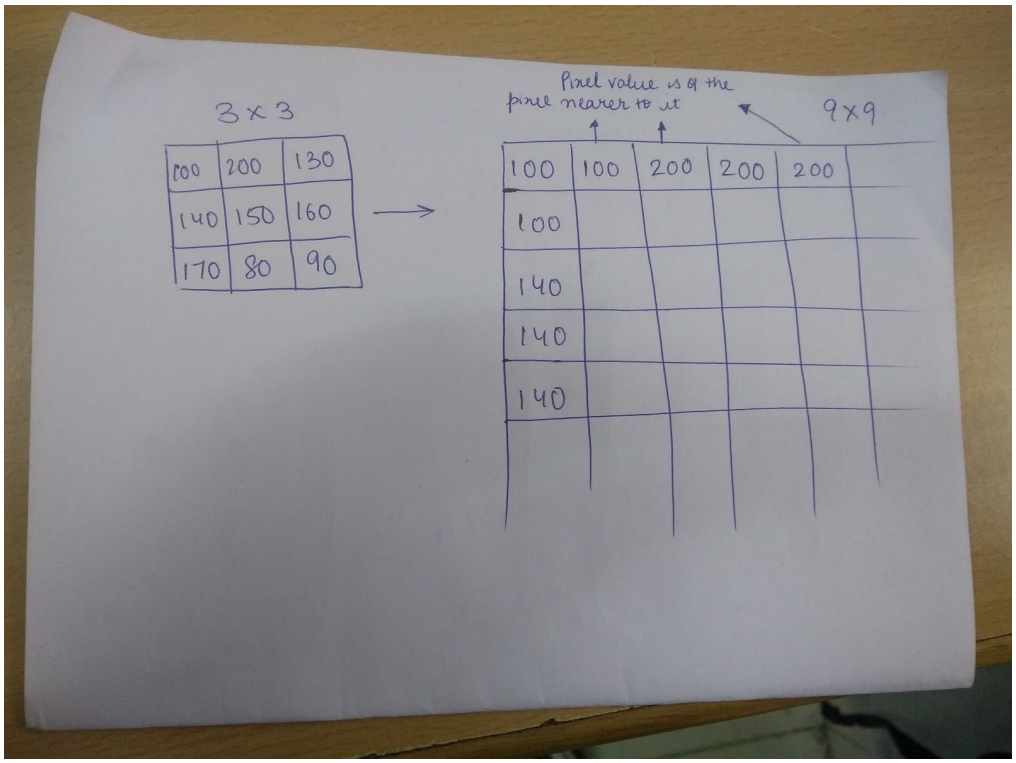
Comparison between Bilinear Interpolation and Nearest Neighbour Interpolation

We see that both fair almost the same.



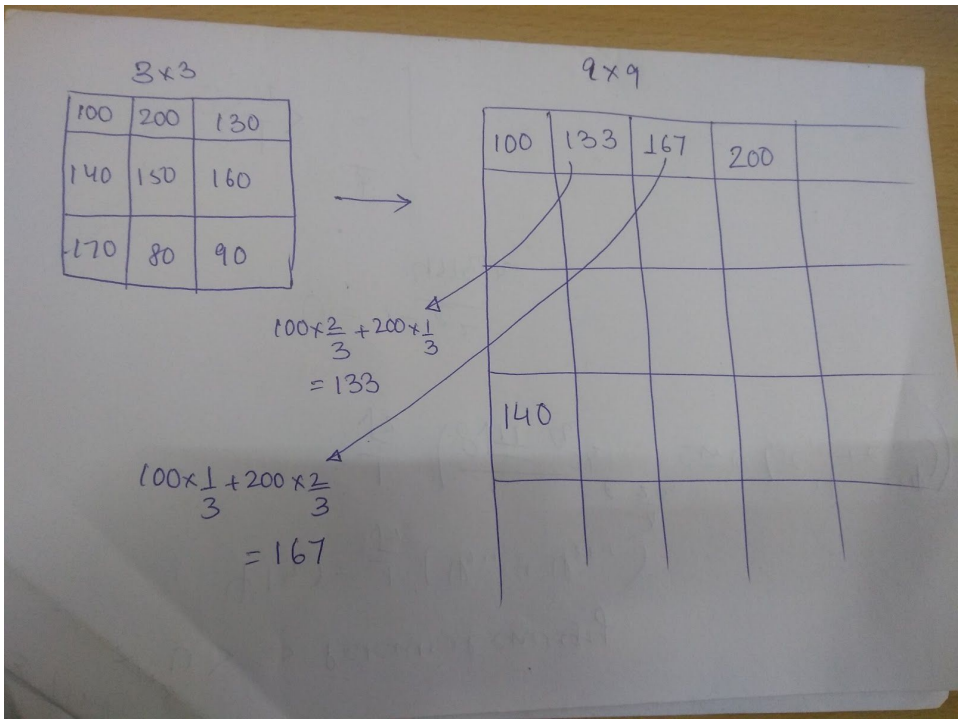
In Nearest Neighbour the pixel value of a pixel's nearest neighbour is copied to it. This may cause abrupt change in pixel values and thus give us pixelated images. This is good for pixelated artwork.

For example: Transform a 3 X 3 matrix to 9 X 9 matrix using Nearest Neighbour



In Bilinear Interpolation we have some smoothness as it's somewhat based on your section formula. The pixel value of the nearest pixel will have more influence on it than the one farther away. But they both contribute to it.

For example: Transform a 3 X 3 matrix to 9 X 9 matrix using Bilinear Interpolation



Nearest Neighbour Interpolation is faster than Bilinear Interpolation.

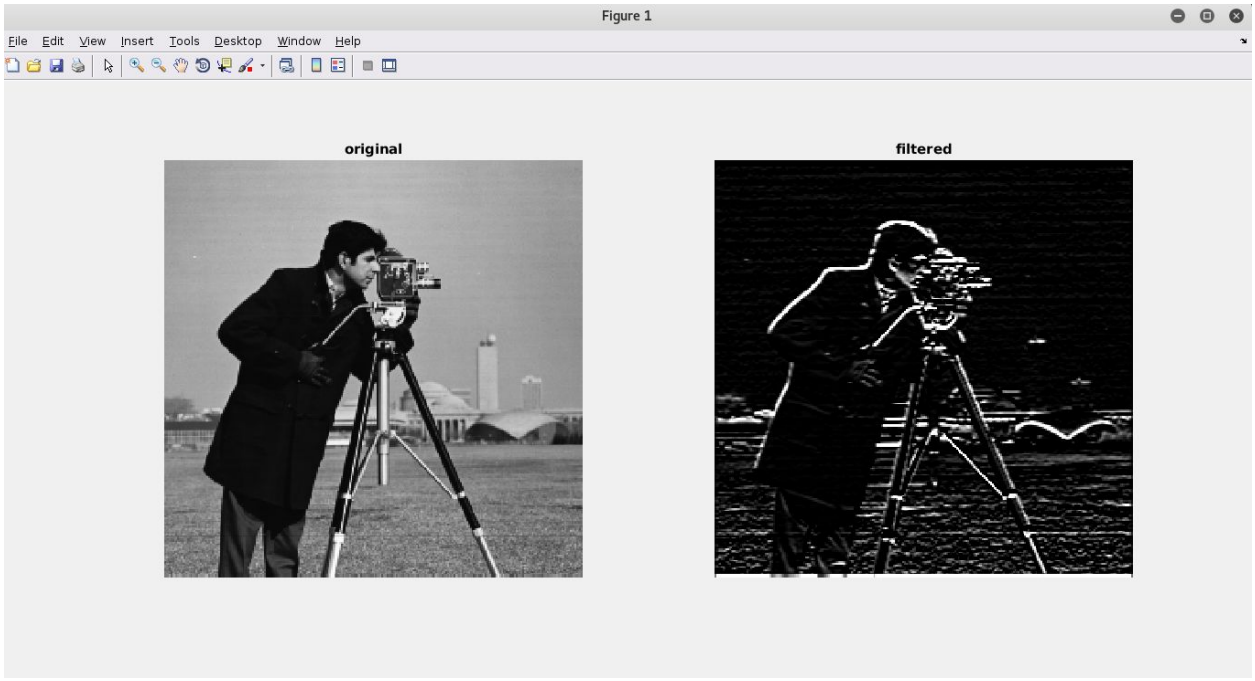
We can use **Bicubic Interpolation** for a better image than Nearest Neighbour or Bilinear Interpolation. It is smoother that both. Basically it uses a cubic equation for finding the pixel values.

Question - 2

The given filter is an Sobel filter.

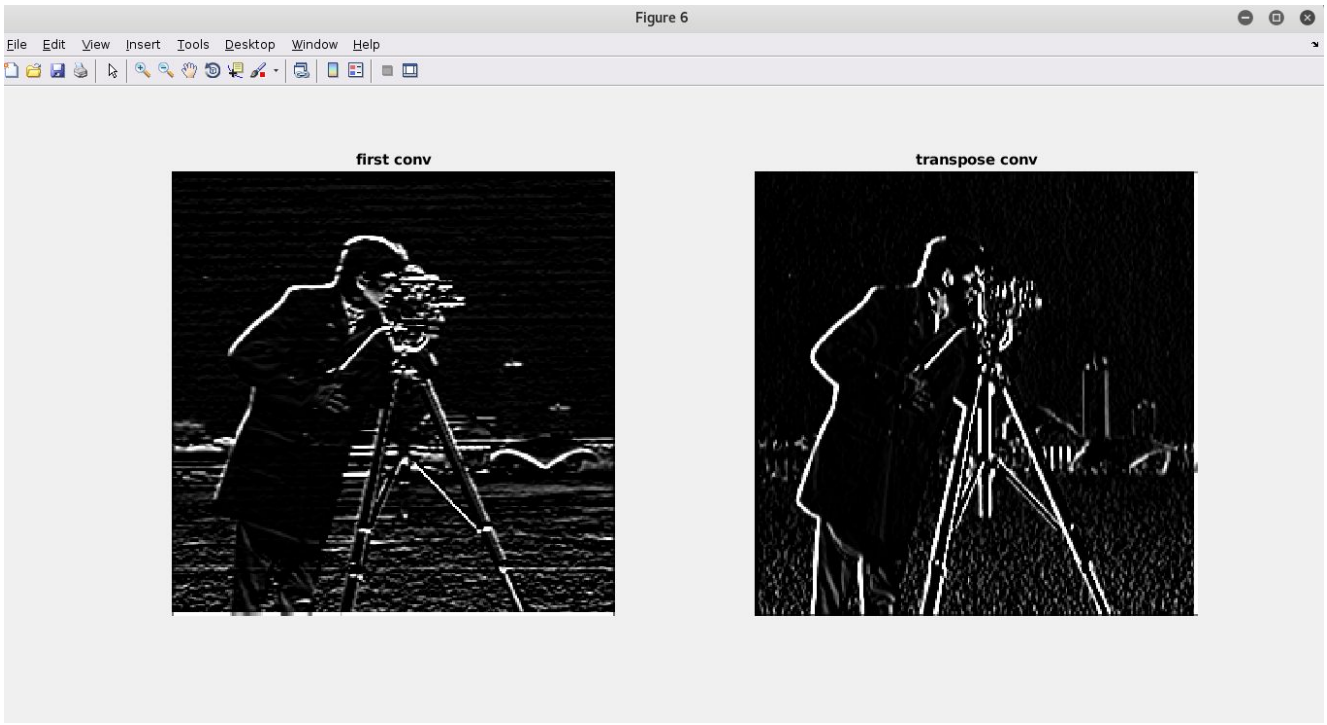
M is a Sobel Filter. The vertical lines are better captured by M^T than by M.

Convolution with M
Detects horizontal edges



Convolution with M^T
Detects vertical edges

Comparison between M and M^T



Question - 3

1. Part 1

output_height = (height + 2 * Z - F)/S + 1

output_width = (width + 2 * Z - F)/S + 1

output_depth = N

2. Part 2

Number of additions = $\sum_{n=0}^{N-1}$ (output_width * output_height) * (F * F - 1) * Channels

Total number of pixels into Multiplication per pixel per channel into number of channels

Number of multiplications = $\sum_{n=0}^{N-1}$ (output_width * output_height) * (F * F) * Channels

Total number of pixels into Multiplication per pixel per channel into number of channels

Question - 4

1. Part 3

Record your own voice using **audiorecorder**. Input is taken for 5 seconds. Save the sound using **audiowrite** and read it using **audioread**.

2. Part 2

Sample-rate conversion is the process of changing the sampling rate of a discrete signal to obtain a new discrete representation of the underlying continuous signal. For more detail about resampling, click on this link.

Now for resampling, the functions used is **linspace**, which generates linearly spaced vectors. It is similar to the colon operator "**:**", but gives direct control over the number of points.

Now in this problem, I have first created an array of the sound of the original sound. Next, I have used linspace to evenly divide this array to the given frequency. And finally used this array to generate the sound.

3. Part 3

Now for this part, we have to simulate the original sound in different environments.

So here, I have performed **resampling** followed by **convolution** with the environment wav file as the filter. The final output is the simulated sound.

Note: We had to make our own resampling function

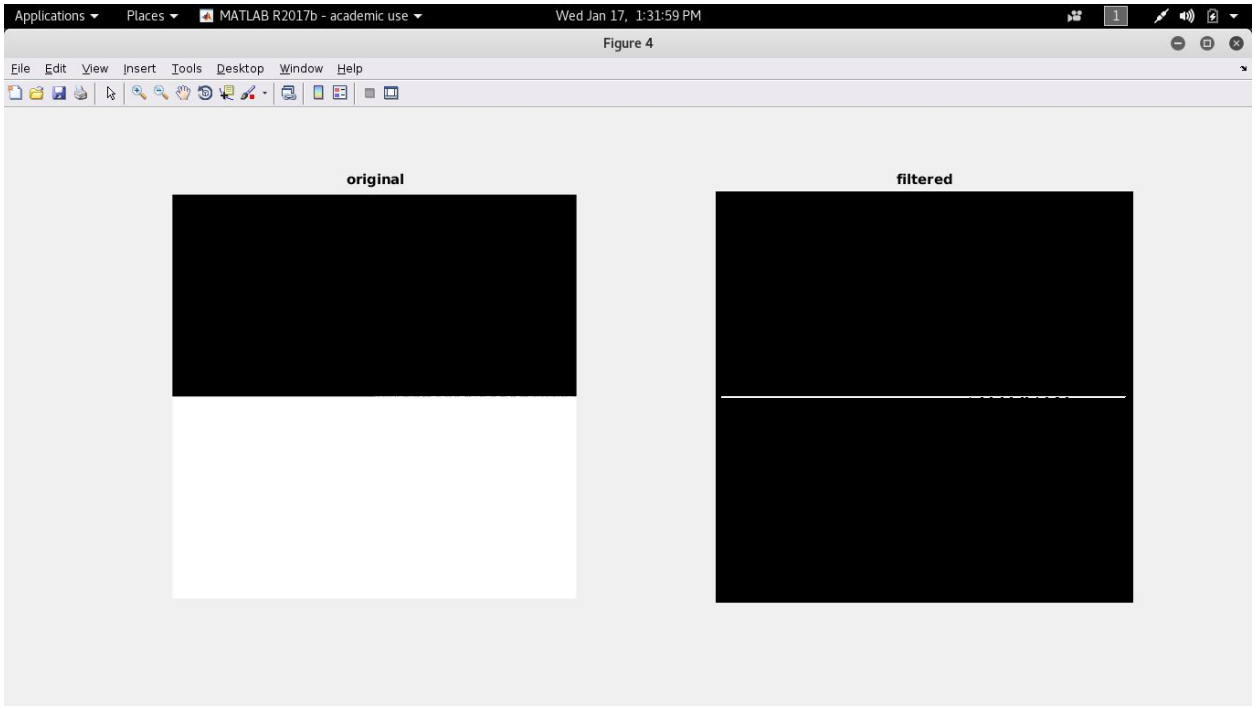
Question - 5

The 3 X 3 Filter is :

0	0	0
1	1	1
-1	-1	-1

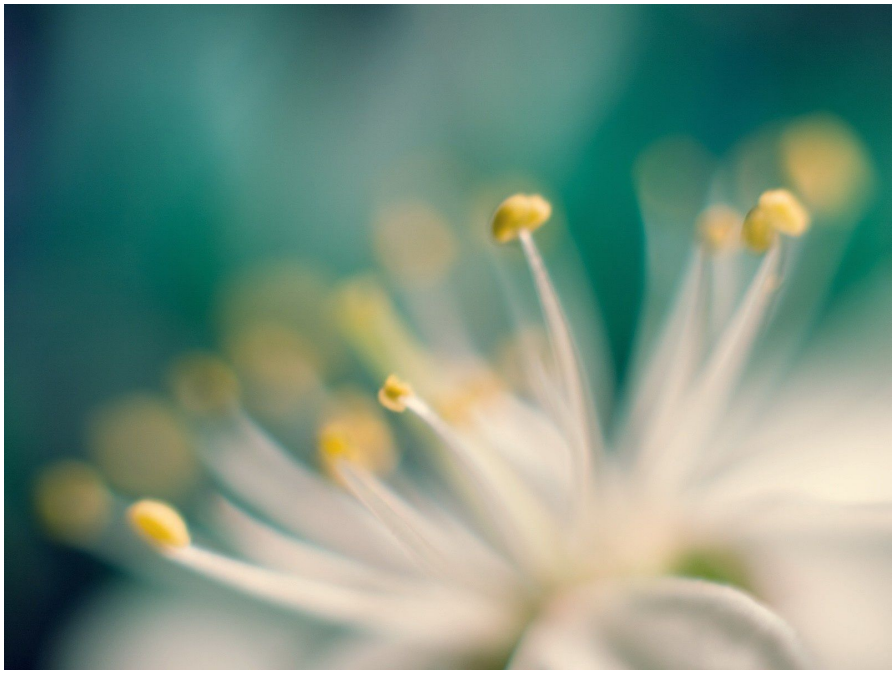
It's an edge detection filter.

Output



The filter and its transpose are applied on blur.jpg

Original



Convolution with filter

blur1.jpg

Convolution with transpose of filter

blur2.jpg

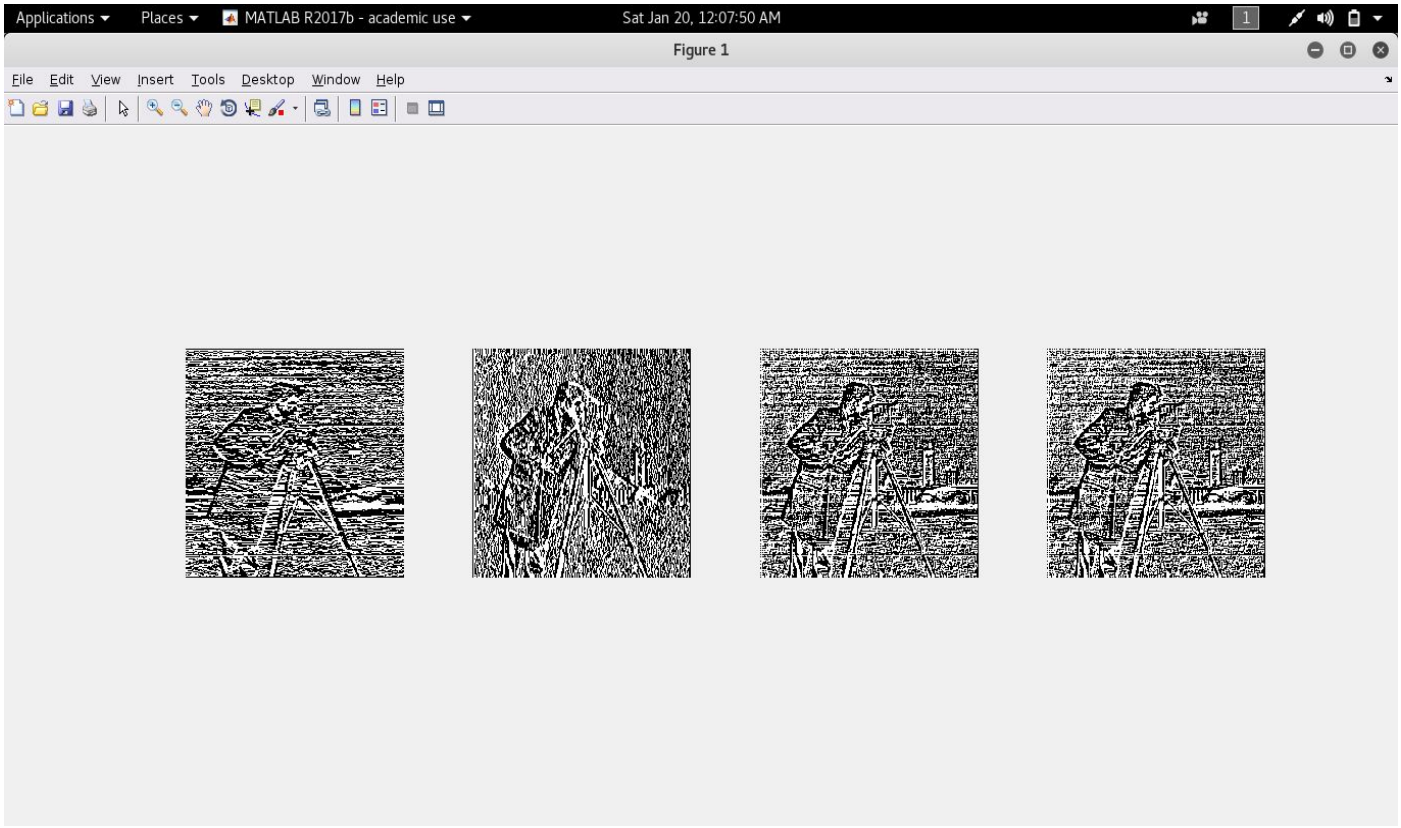
Adding blur1.jpg and blur2.jpg

Adding blur.jpg, blur1.jpg and blur2.jpg

3 more images

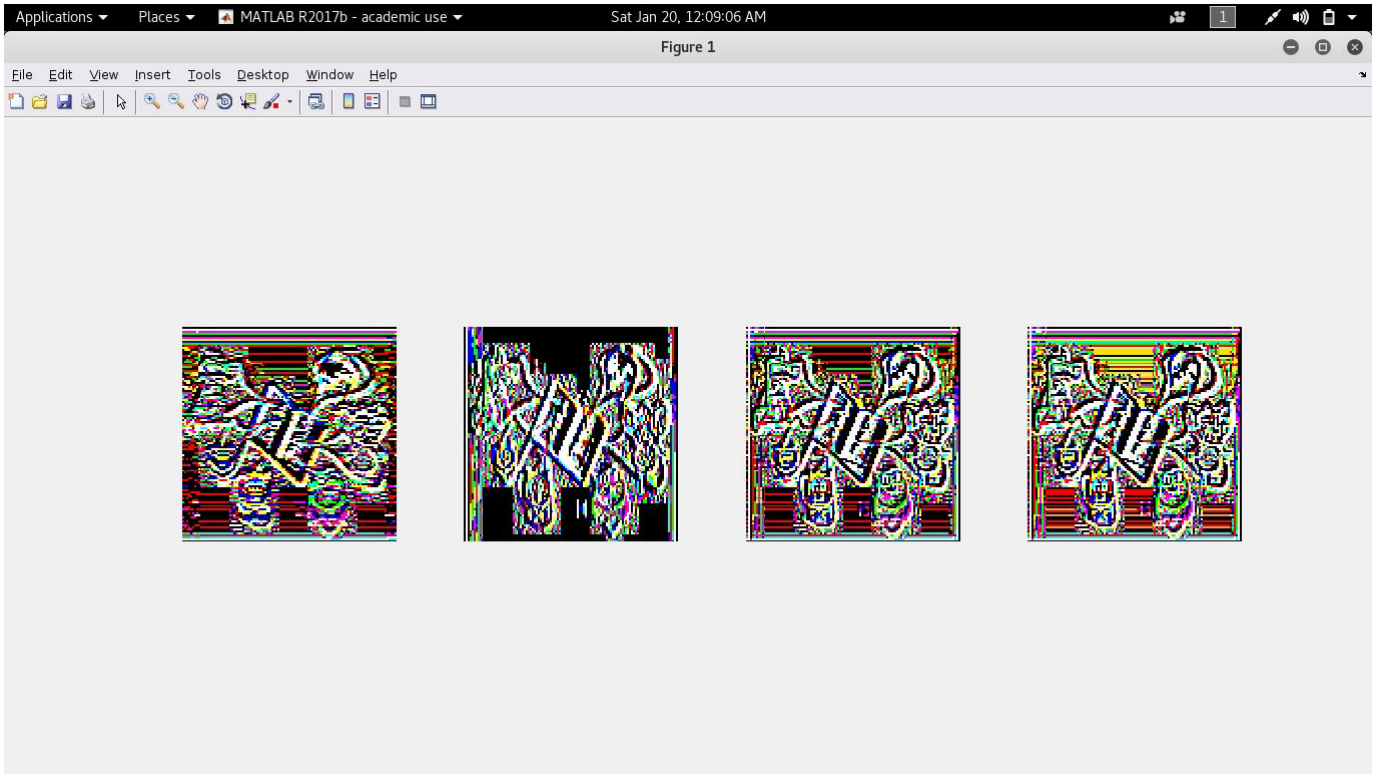
Original

Using filter and its transpose and adding them up together and with the original image



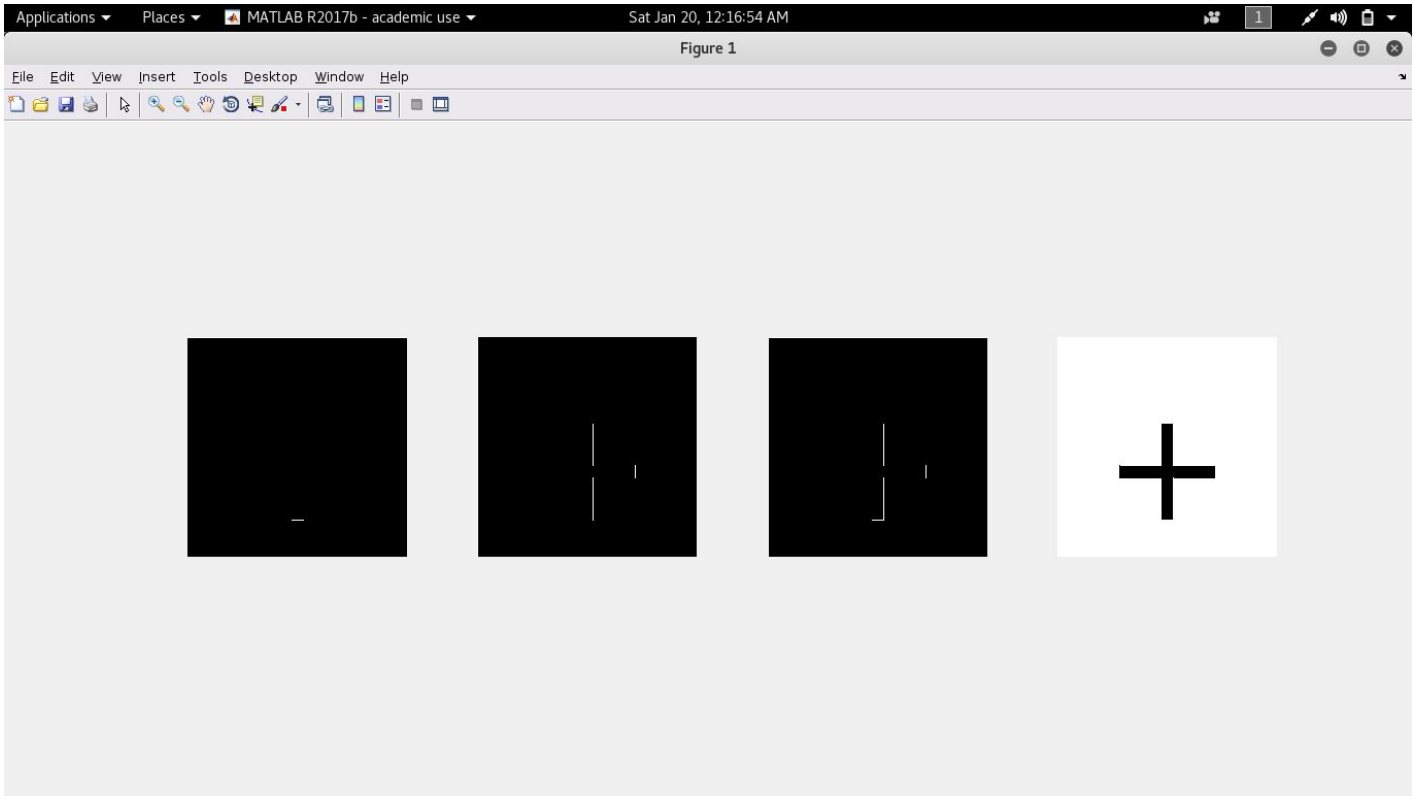
Original

Using filter and its transpose and adding them up up together and with the original image



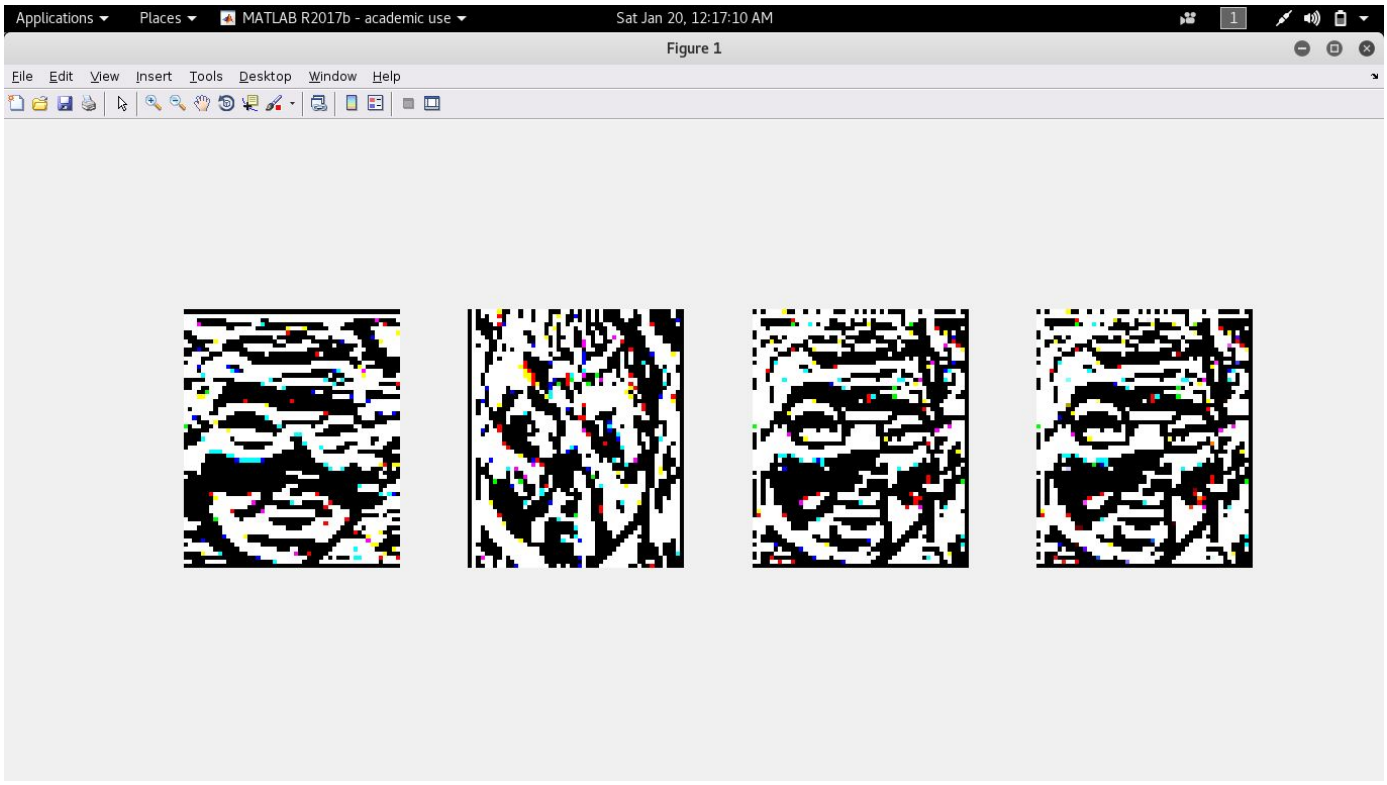
Original

Using filter and its transpose and adding them up up together and with the original image


Applications
Places
MATLAB R2017b - academic use
Sat Jan 20, 12:16:54 AM
Figure 1
File Edit View Insert Tools Desktop Window Help

Original

Using filter and its transpose and adding them up up together and with the original image



M detects the horizontal edges whereas M^T detects vertical edges.

Question - 6

Finding a sub image from a parent image by traversing and checking. Just take the size of the sib image from parent image and find how many pixels match. Where the maximum pixels match that is the sub image.

Note: We are not allowed to use normxcorr2

Sub Image

Parent Image

Result

Question - 7

We have to find out the filter matrix in this problem.
Firstly we have to find out the length of the filter matrix using the formula:

Filter_length = Input_length - Output_length + 1

Note: No padding is assumed and stride assumed is 1

Flipping is done only once as it's a 1D signal

I have used the sliding array technique to form the equations for the elements of the filter and using these equations, created a matrix. Then using the matrix manipulations, we found out the filter matrix.