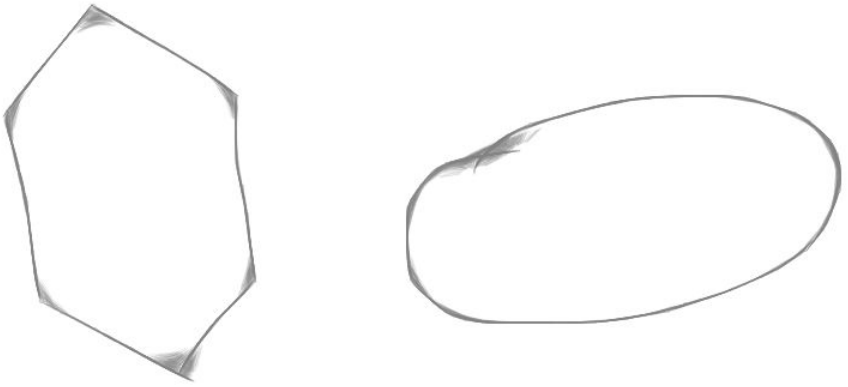
ASSIGNMENT 1

NAME - AKSHAT MAHESHWARI
ROLL NO. - 20161024

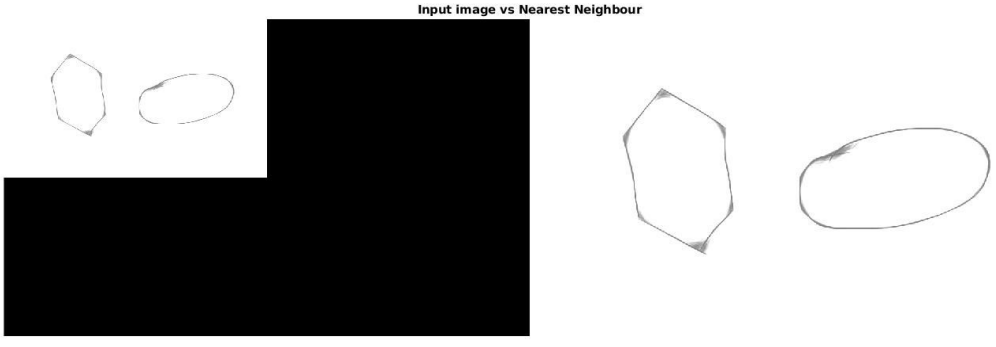
QUESTION - 1

a. X = 2

1. Original

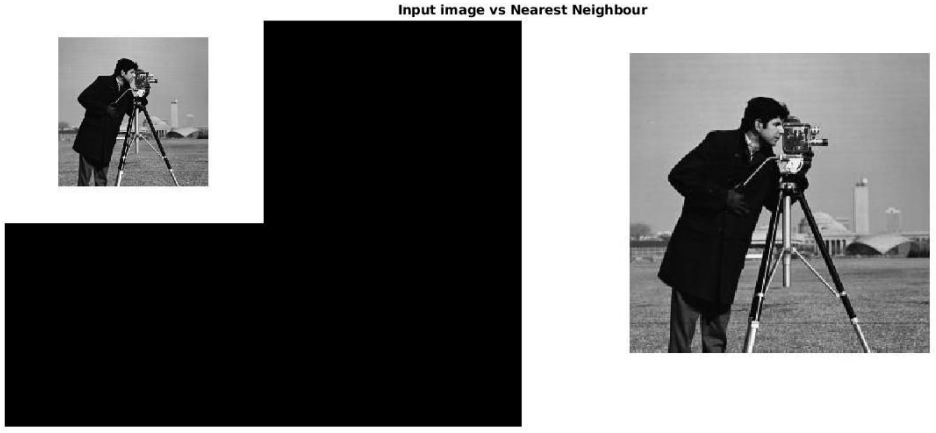


On Expansion (Nearest Neighbour Interpolation)



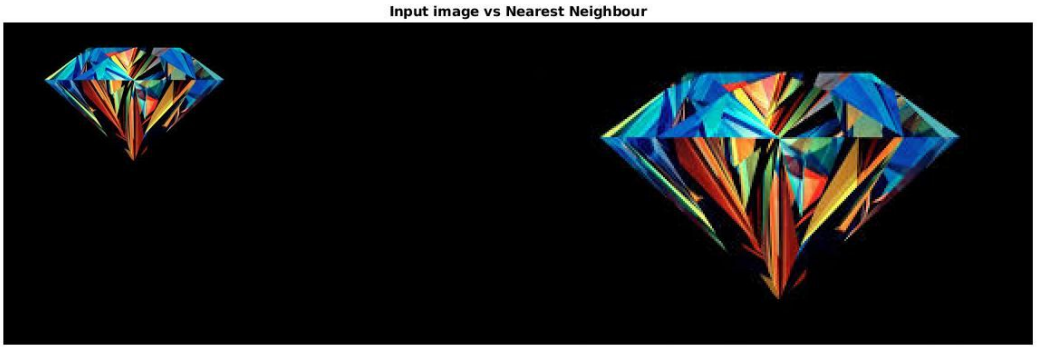
2. Original

On Expansion (Nearest Neighbour Interpolation)



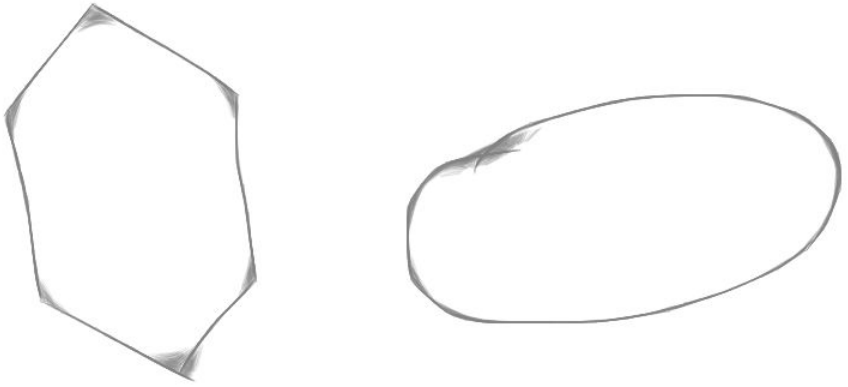
3. Original

On Expansion (Nearest Neighbour Interpolation)

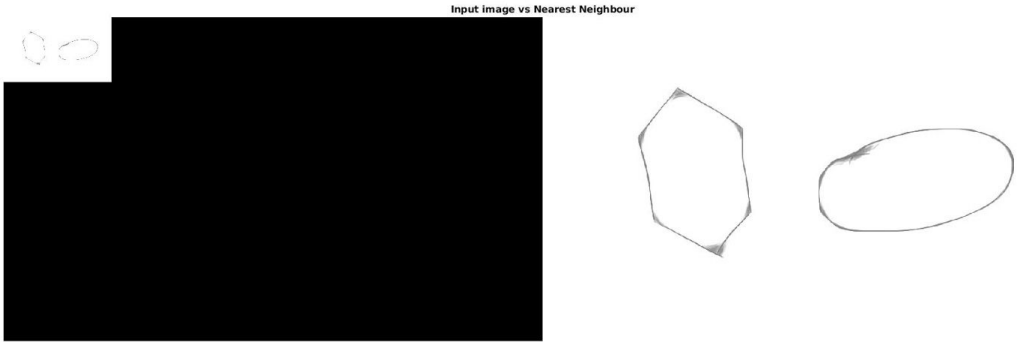


b. X = 5

1. Original



On Expansion (Nearest Neighbour Interpolation)

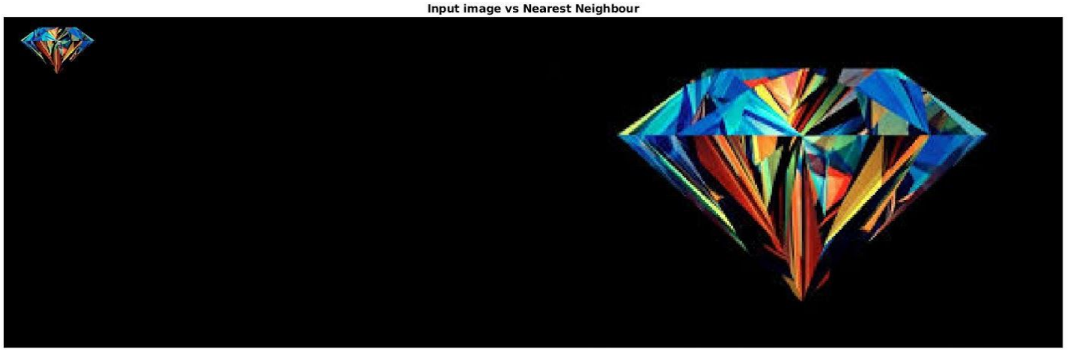


2. Original

On Expansion (Nearest Neighbour Interpolation)

3. Original

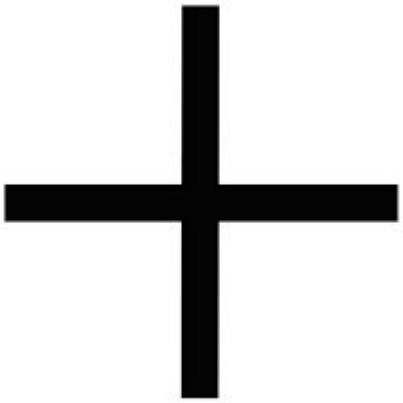
On Expansion (Nearest Neighbour Interpolation)



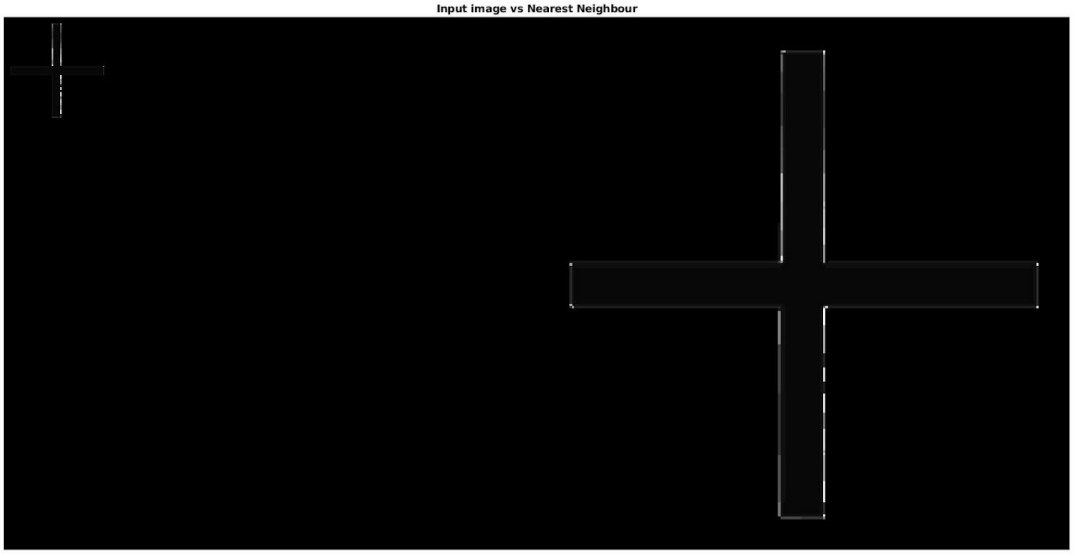
We see that the image formed by **Nearest Neighbour Interpolation** is more pixelated that the **Bilinear Interpolation** in maximum cases. Nearest Neighbour fairs equally or even better with images containing high number straight edges.

- For example:

  Original

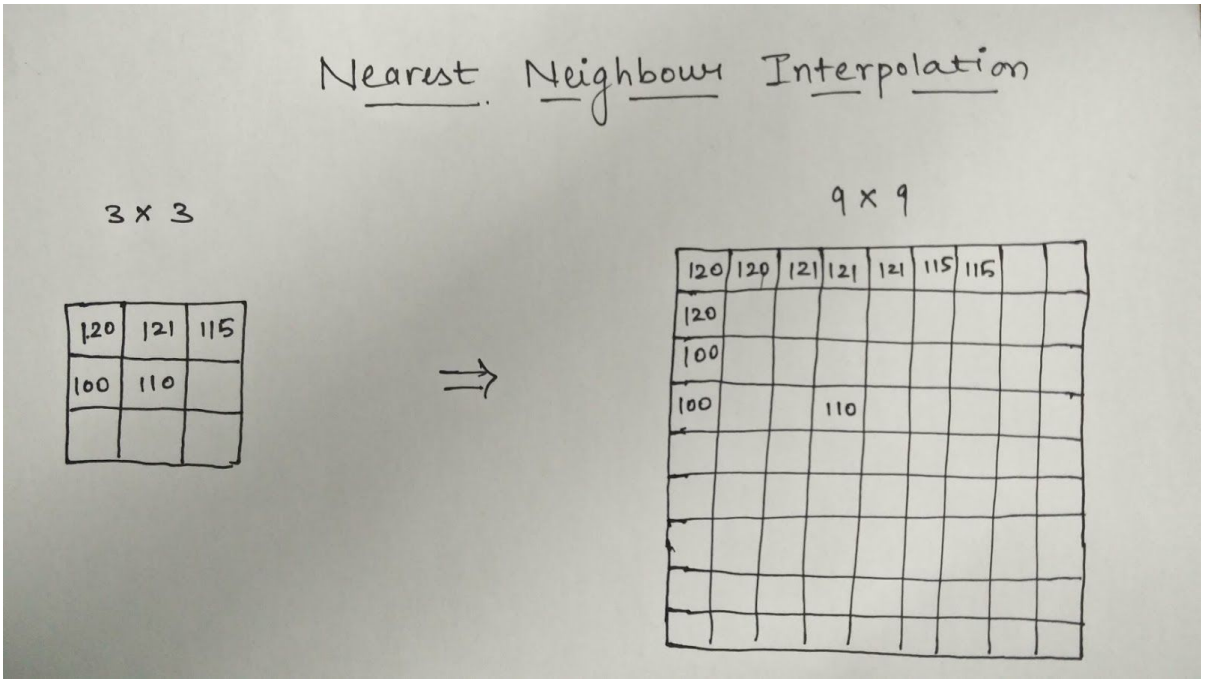


  X = 5 (Nearest Neighbour Interpolation)



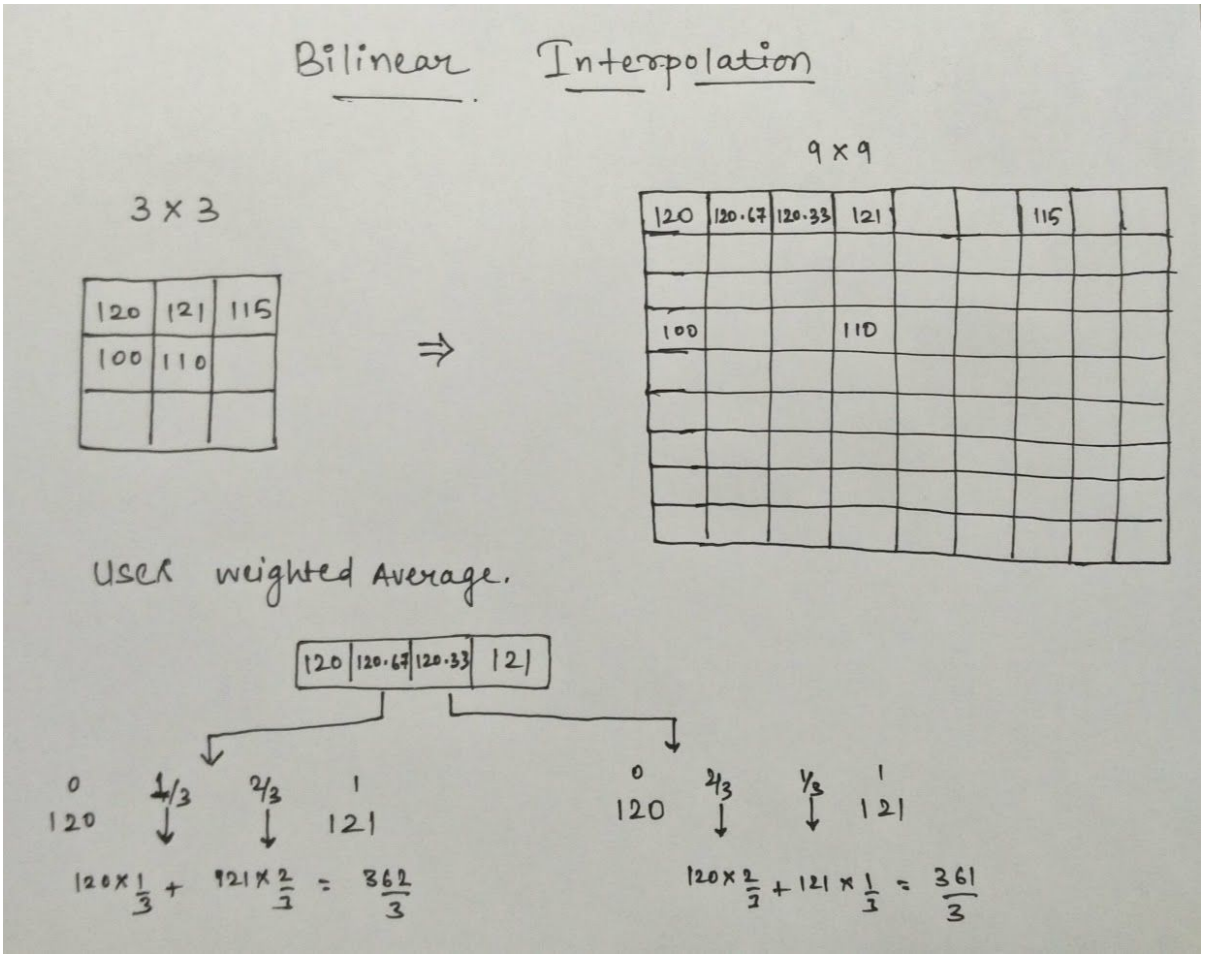
In Nearest Neighbour the pixel value of a pixel's nearest neighbour is copied to it. This may cause abrupt change in pixel values and thus give us pixelated images. This is good for pixelated artwork.

- 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 embossing filter.

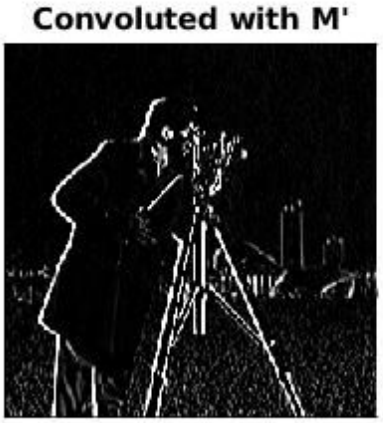
M causes embossing. It also shows clearer picture of background as compared to $\mathbf{M^T}$ **(M Transpose)**. M' shows clearer edges for the Cameraman in foreground. The vertical lines are better captured by M' than by M.

- Convolution with M

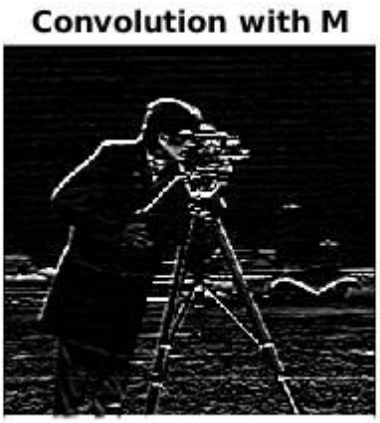
Original

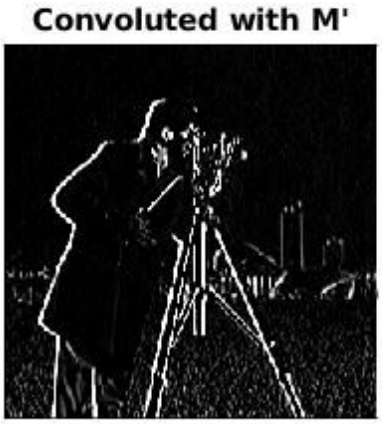
Convoluted with M

- Convolution with M^T



- Comparision between M and M^T Convolution





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 = (F * F - 1) * Output_Width * Output_Height*

   *Number of Multiplications = F * F * Output_Width * Output_Height*

QUESTION - 4

1. Part 1

   Recording our own voice was done using the in-built function ***audiorecorder***, followed by ***recordblocking***, which basically records the sound for a particular time which we give as input (here I have taken the time to be 5 sec.).

Next to save the sound, the ***audiowrite*** function was used. And similarly, to read the file, I have used ***audioread*** function, which is built-in.

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, I have used some functions like ***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.

QUESTION - 5

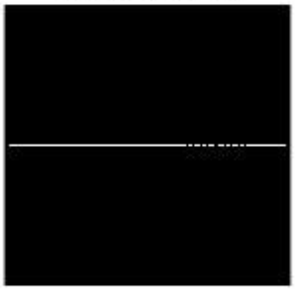
The **3 X 3 Filter** is :

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

It's an **Line detecting filter**.

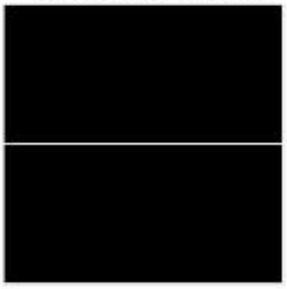
- Output

My Output



Sample Input

Sample Output



- Final Outputs of all operations on blur

Convolution with original

Convolution with transpose

Original blur.jpg

Final addition image

- 3 more images:
    1. Appling all operations on cameraman.jpg

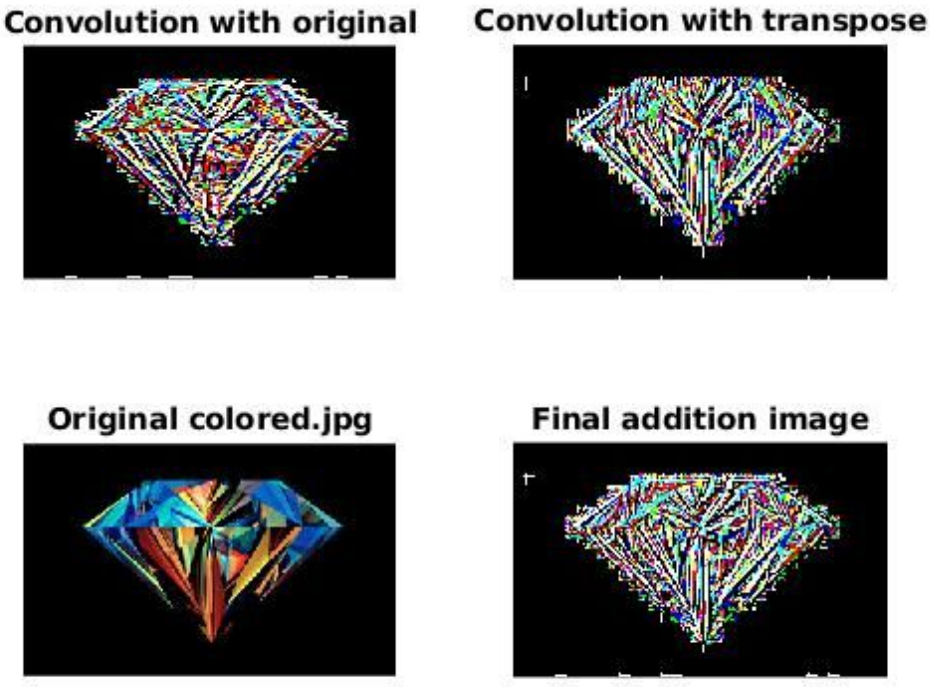
Convolution with original

Convolution with transpose

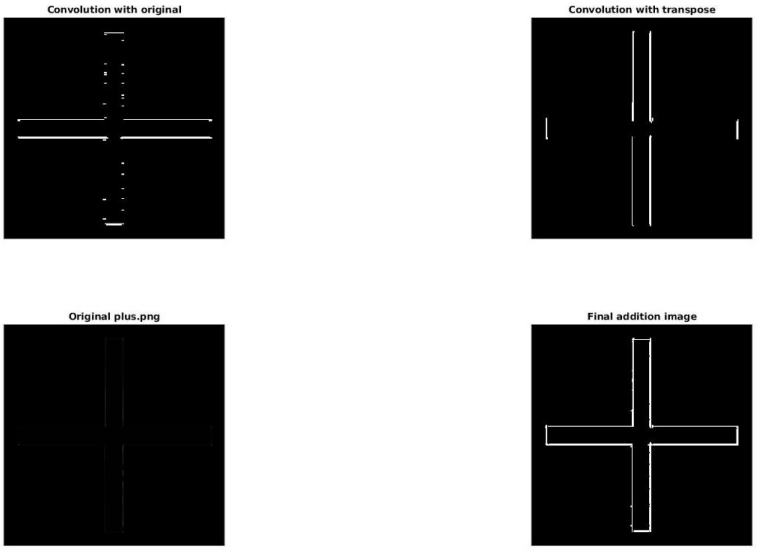
Original cameraman.tif

Final addition image

2. Applying all operations on colored.jpg



3. Applying all oprations on plus.jpg



M detects the **horizontal edges** whereas $\mathbf{M}^T$ detects **vertical edges**.

QUESTION - 6

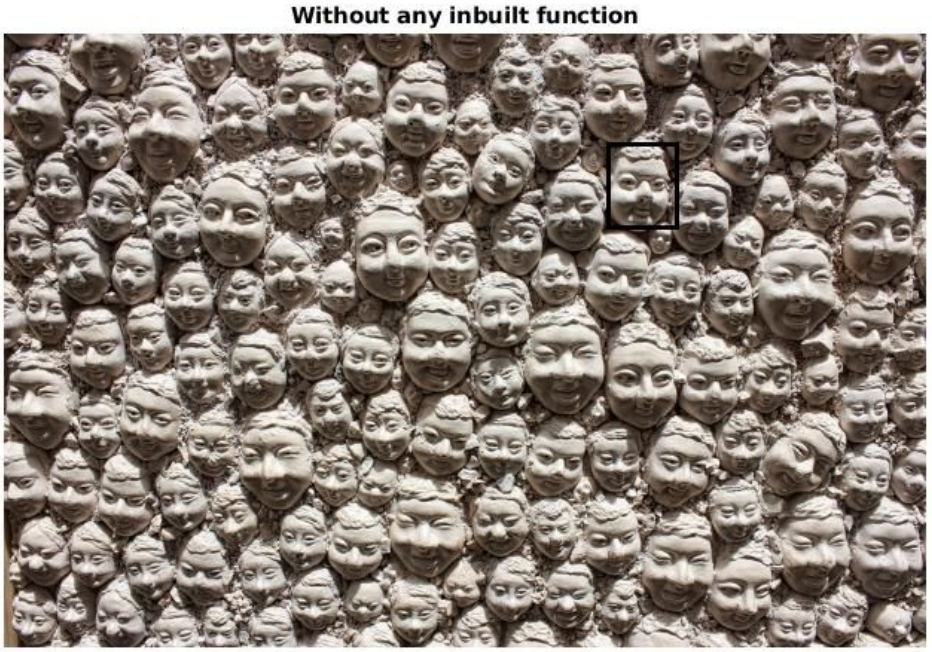
Finding a sub image from a parent image by traversing and checking.

Note: We are not allowed to use normxcorr2

- Sub Image

- Parent Image

- Result



By default, matlab has inbuilt function of normxcorr2 (Normalized Cross-Correlation for 2D matrix).
normxcorr2 uses the following general procedure:

1. Calculate cross-correlation in the spatial or the frequency domain, depending on size of images.
2. Calculate local sums by precomputing running sums.
3. Use local sums to normalize the cross-correlation to get correlation coefficients.

The implementation closely follows the formula from:

$$\gamma(u,v) = \frac{\sum_{x,y}\left[f(x,y) - \overline{f}_{u,v}\right]\left[t(x-u,y-v) - \overline{t}\right]}{\left\{\sum_{x,y}\left[f(x,y) - \overline{f}_{u,v}\right]^2 \sum_{x,y}\left[t(x-u,y-v) - \overline{t}\right]^2\right\}^{0.5}}$$

where

- f is the image.
- $\overline{t}$ is the mean of the template
- $\overline{f}_{u,v}$ is the mean of f(x,y)in the region under the template.

QUESTION - 7

We have to find out the filter matrix in this problem.

So I have done something similar to **deconvolution**.

Firstly I have found out the length of the filter matrix using the formula:

$$\textbf{\textit{Filter\_length = Input\_length - Output\_length + 1}}$$

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, I have found out the filter matrix.

Note: I have assumed zero padding and stride length to be 1.