

EE-569

INTRODUCTION TO
DIGITAL IMAGE
PROCESSING

HOMEWORK #2

DIXITH REDDY GOMARI

USC ID-3098766483

gomari@usc.edu

Problem 1: Geometric Image Modification

a) Geometrical Warping

1.1 Motivation:

- Geometrical warping of images is a popular technique to modify images on the basis of translation, rotation and scaling.
- This technique is popularly used in computer graphics and several other medical image processing applications.
- In this assignment, the objective was to match the give image to the required geometrically warped image.

1.2 Approach and Procedure:

- I have performed the geometrical warping by linear interpolation of the image.
- For this procedure I have divided my original image into four parts of equal sizes and then performed the linear interpolation on all these parts separately.
- Let's consider the first part, the scaling factor for each row varies as the no of points to be mapped to varies according to the row.
- If $F(p,q')$ is the intensity of the pixel of the output image at position p,q' and let us say we are resizing a row of size a to size b , then by reverse mapping $F(p,q')$ will be present at and $(a/b)*q'$ position of the input image.
- As decimal value positions do not exist in a digital image, in order to find the intensity value, we need to perform linear interpolation technique.
- Let us say the reverse mapped pixel exists between 2 coordinates of the input image say $F(p,q)$, $F(p,q+1)$ then $\Delta y = ((b/d) * q') - q$.
- Then $F(p,q')$ is given by
$$F(p, q') = (1 - \Delta y)F(p, q) + \Delta yF(p, q + 1).$$
- As we can see there is only difference in the mapping of the column but there is no change in the row value as it is a linearly mapped image.

1.3 Results:



Figure 1 Linear mapping for the first quadrant



Figure 2 For 2nd Quadrant



Figure 3 For 3rd Quadrant



Figure 4 Final warped Image



Figure 5 Linear mapping for the first quadrant

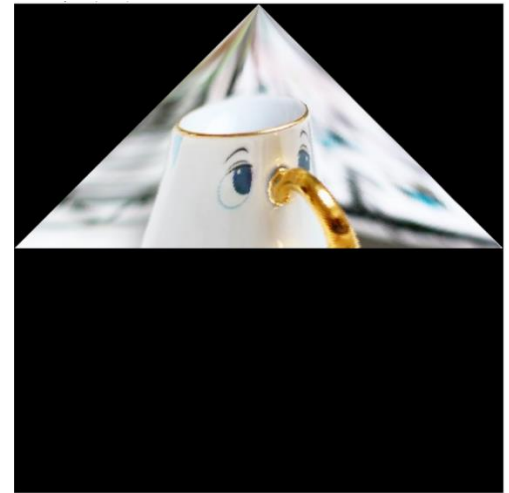


Figure 6 For 2nd Quadrant



Figure 7 For 3rd Quadrant



Figure 8 Final warped Image

1.4 Discussion

- I have performed linear warping method to achieve the required diamond shape from a square shaped image.
- The output image is a little distorted, as we are trying to down sample each row to lower sizes.
- In this procedure, all the pixel intensities are not maintained which results in the loss of information.
- The edges are also preserved in the warped image but the transition from the input image to output image is not very smooth.

(b) Puzzle Matching

1.5 Motivation:

- Puzzle matching can be obtained by cascading translation, rotation and scaling matrices for the puzzle piece.
- It can be obtained by affine projection also by mapping the piece coordinates to the required position on the image.
- The puzzle matching techniques can be used in computer graphics and medical image processing where we can match the missing places with puzzle pieces.

1.6 Approach and Procedure:

- Initially, I have separated the puzzle pieces and worked on them separately.
- After separating the corresponding puzzle pieces, I have found the corresponding four corners of the puzzle pieces.
- Calculate the rotation angle theta with the coordinates of the four corners.

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

- Compute rotation on all the pixels by plugging in the value of theta in the rotation matrix which is given as follows,

$$\begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_i \\ 1 \end{bmatrix}$$

- Once the image is rotated, I have calculated the four corners of the image again and translated it to the top corner of the image coordinates.
- As the size of the puzzle piece does not match the size of the missing piece we need to perform scaling operation on the puzzle piece.
- For this procedure, find the scaling factor of x and y coordinates S_x and S_y and plug these values in the scaling transformation matrix and scale each and every pixel of the puzzle piece. This can be performed as follows,

$$\begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & 0 & 0 \\ 0 & \frac{1}{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_i \\ 1 \end{bmatrix}$$

- As we cannot have decimal image pixel positions in digital images, we need to perform bilinear interpolation to get the desired value.

$$\hat{F}(p, q) = (1 - b)[(1 - a)F(p, q) + aF(p + 1, q)] + b[(1 - a)F(p, q + 1) + aF(p + 1, q + 1)]$$

- Now, we have to find the missing coordinates in our corresponding image, once this is done just copy the corresponding values into our desired image.

1.7 Results

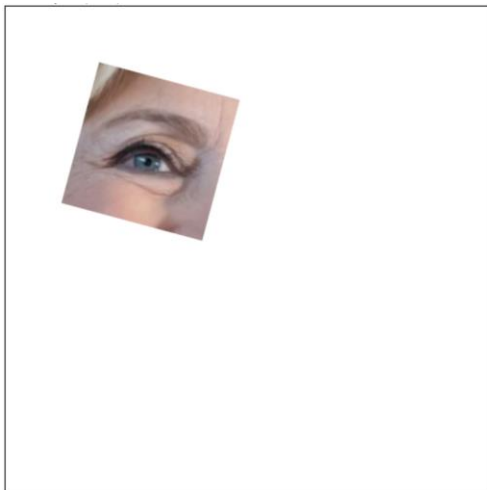


Figure 9 Separated Image

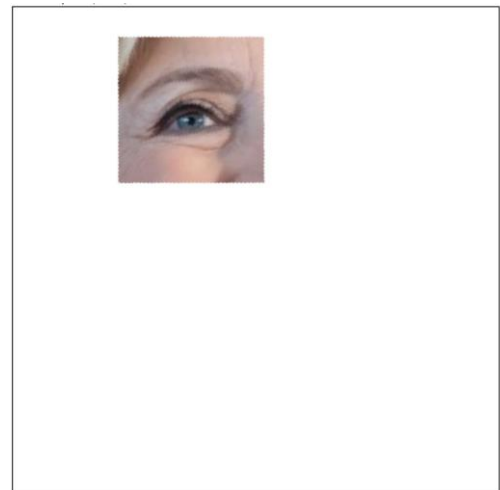


Figure 10 Rotated Image

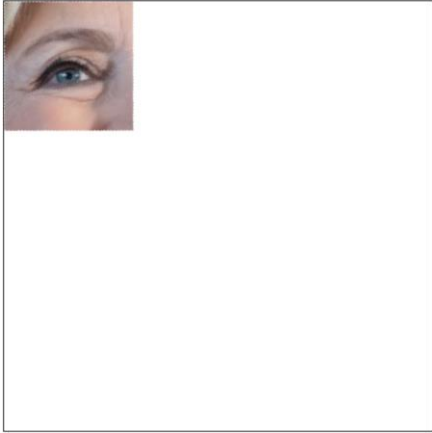


Figure 11 Translated Image

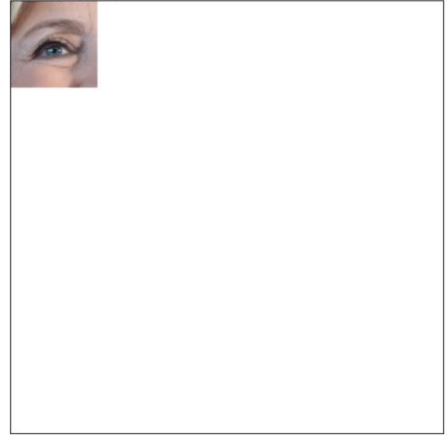


Figure 12 Scaled Image



Figure 13 Final Hillary Image

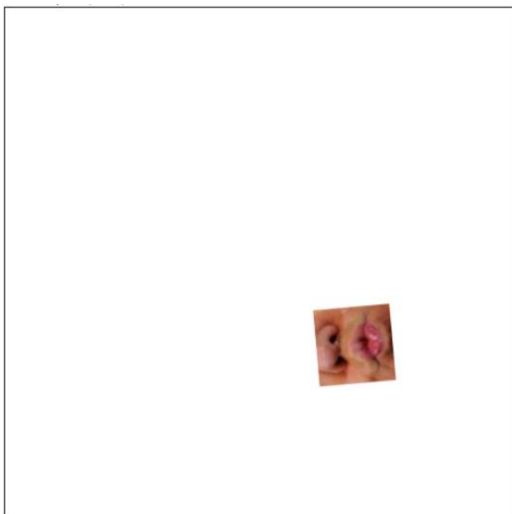


Figure 14 Separated Image

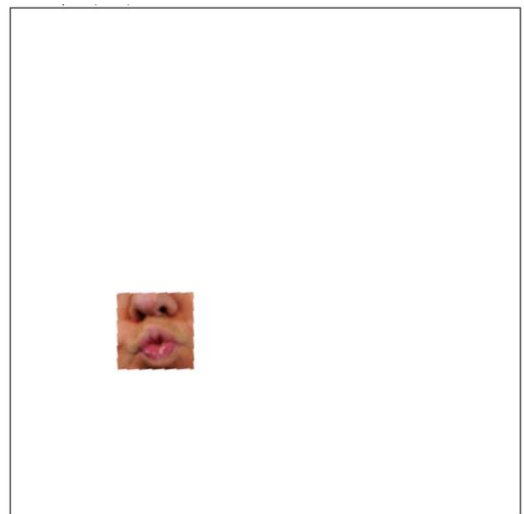


Figure 15 Rotated Image

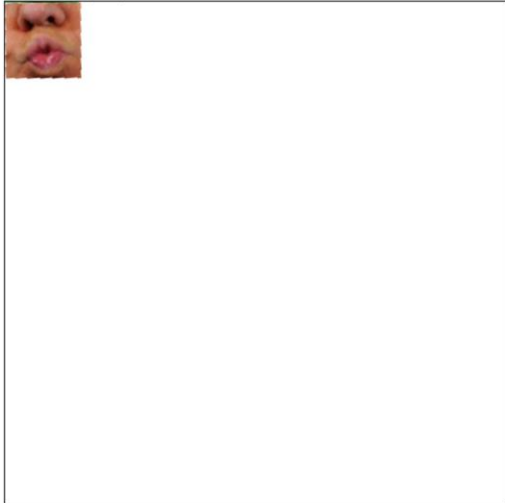


Figure 16 Translated Image



Figure 17 Scaled Image

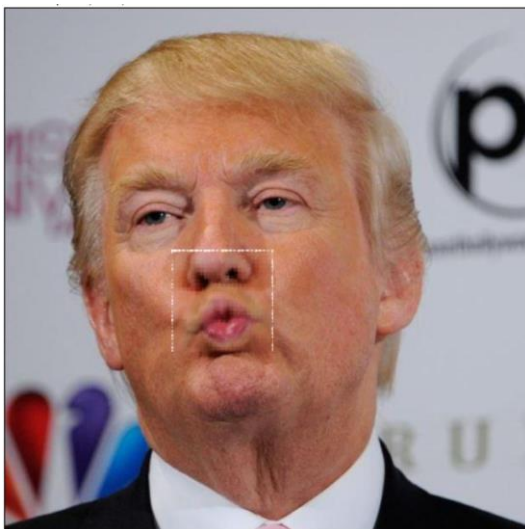


Figure 18 Problematic borders



Figure 19 Final Image

1.8 Discussion

- As we can see in figure 18, the borders of the puzzle aren't efficient.
- This problem was not evident in Hillary as we were down sampling while scaling which compensates this effect
- I have reduced this effect by averaging the pixels near the border of the mouth of the trump image.
- This can also be tackled by scaling the puzzle piece to a slightly larger value than required and then copying the corresponding pixel value.

(c) Homographic Transformation and Image Overlay

1.9 Motivation:

- Homograph Transformation is a popular technique used in geo-referencing.
- The perspective to plane transformation can be performed by homographic transformation i.e., to generate an object based on a perspective photo.
- These techniques are generally used in image manipulation to achieve perspective viewing.
- It involves projection from 3D to 2D camera co-ordinates.

1.10 Approach and Procedure:

- In homographic transformation, I followed reverse mapping technique to achieve the desired output.
- In the initial step we need to find the coordinates of the field image that we want to map to.
- Convert those coordinates into Cartesian form.
- Find the Cartesian coordinates of the Trojan image.
- Let's say $(x1,y1)$, $(x2,y2)$, $(x3,y3)$, $(x4,y4)$ are the Cartesian coordinates of the field image and $(X1,Y1)$, $(X2,Y2)$, $(X3,Y3)$, $(X4,Y4)$ are the Cartesian coordinates of the Trojans image.
- By using the above coordinates generate the Homographic matrix. I have generated this in MATLAB.

```
x1=320.5; x2=542.5; x3=653.5; x4=528.5;
y1=50.5; y2=220.5; y3=217.5; y4=34.5;
X1=0.5; X2=349.5; X3=349.5; X4=0.5;
Y1=196.5; Y2=196.5; Y3=0.5; Y4=0.5;
a=[x1 y1 1 0 0 0 (-x1*X1) (-y1*X1);...
   x2 y2 1 0 0 0 (-x2*X2) (-y2*X2);...
   x3 y3 1 0 0 0 (-x3*X3) (-y3*X3);...
   x4 y4 1 0 0 0 (-x4*X4) (-y4*X4);...
   0 0 0 x1 y1 1 (-x1*Y1) (-y1*Y1);...
   0 0 0 x2 y2 1 (-x2*Y2) (-y2*Y2);...
   0 0 0 x3 y3 1 (-x3*Y3) (-y3*Y3);...
   0 0 0 x4 y4 1 (-x4*Y4) (-y4*Y4)];
b=[X1;X2;X3;X4;Y1;Y2;Y3;Y4];
h=inv(a)*b
```

- The H matrix is given by

h =

0.0657
0.8522
-63.6415
-0.7932
0.5405
401.0036
0.0001
-0.0026

- Once we get the Homographic matrix, we need to find the equations of line in which we need to traverse in the field image.
- Plug those values in the formula below,

$$\begin{bmatrix} x'_2 \\ y'_2 \\ w'_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x'_2}{w'_2} \\ \frac{y'_2}{w'_2} \end{bmatrix}$$

Where x_2, y_2 are the Cartesian coordinates of the Trojans image and x_1 and y_1 are the coordinates of the field image.

- Once the image is projected, remove the white background from the Trojans image.

1.11 Results



Figure 20 Original Field image



Figure 21 Original Trojans image



Figure 22 Homographic transformed image

1.12 Discussion:

- As we can see that an image can be projected on any other image when given the coordinates to be projected on.
- Even though the image is projected onto it, it is covering all the lines of the field image.
- For a better perspective of the two images, there are techniques to improve the image.
- One method we can use is layer blending technique to improve the results of the above image.
- The result can be seen below



Figure 23 Improved Result after Layer Blending

Problem 2: Digital Halftoning

(a) Dithering Matrix

2.1 Motivation:

- Halftoning is a method of creating illusions of continuous tone output with the help of just binary colors.
- Digital Halftoning is generally used in newspapers to recreate images with lower cost by just using different tones of black dots.
- Halftoning helps to reduce the cost of ink to reproduce images.
- Effective Digital Halftoning can improve the quality of recreated images at minimal cost.

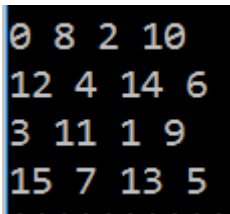
2.2 Approach and Procedure:

- There are many ways to perform dithering on grayscale images. Here we are using Bayer's index matrix to perform Digital Halftoning.
- Bayer's Matrix indicates the pixel that is more likely to be turned on.
Given I_2 matrix, i.e.,

$$I_2(i, j) = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

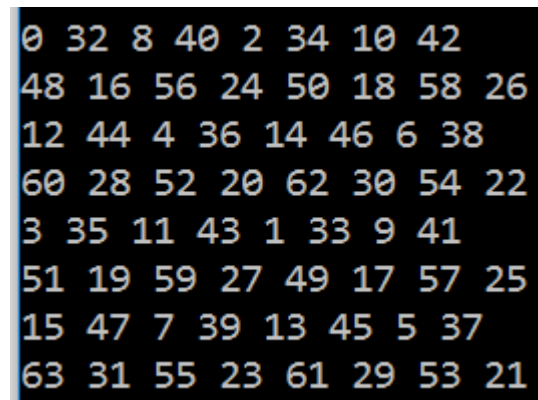
- I have generated I_4 and I_8 matrix with the below formula

$$I_{2n}(i, j) = \begin{bmatrix} 4 * I_n(x, y) & 4 * I_n(x, y) + 2 \\ 4 * I_n(x, y) + 3 & 4 * I_n(x, y) + 1 \end{bmatrix}$$



0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

Figure 1 I_4 Matrix



0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

Figure 2 I_8 Matrix

- Now compute the corresponding threshold matrix for the respective Bayer matrix by,

$$T(x, y) = \frac{I(x, y) + 0.5}{N^2} \times 255$$

Where N^2 is the size of the Bayer Matrix.

- Now turn the pixel on or off by comparing it with the threshold, the computation involved in this operation is as follows,

$$G(i, j) = \begin{cases} 1 & \text{if } F(i, j) > T(i \bmod N, j \bmod N) \\ 0 & \text{otherwise} \end{cases}$$

Where $G(i, j)$ and $F(i, j)$ are the normalized values of the output and input respectively.

- There are other ways of doing Digital Halftoning. One method is by the use of the below matrix,

$$A_4(i, j) = \begin{bmatrix} 14 & 10 & 11 & 15 \\ 9 & 3 & 0 & 4 \\ 8 & 2 & 1 & 5 \\ 13 & 7 & 6 & 12 \end{bmatrix}$$

- Digital Halftoning for this matrix is performed similarly to that of I4 matrix, instead just use the weights of the A_4 matrix.
- The Digitally Half toned images can be see below.

2.3 Results:



Figure 3 Original Man Image



Figure 4 12 Dithered Image



Figure 5 18 Dithered Matrix



Figure 6 14 Dithered Matrix



Figure 7 44 Dithered Matrix



Figure 8 Image recreated with four intensities

2.4 Discussion

- We can clearly make out the difference between different dithered matrix.
- As the size of the matrix increases the probability of pixel to be turned on.
- We can see that number of pixels that are turned on increases as the size of matrix increases in the above figures.
- While coming to A4 matrix we can see some grid patterns in the Dithered image.
- This technique is called Clustered dot screens method i.e., if the consecutive thresholds are located in spatial proximity, then it is called “Clustered Dot Screen”.

- I would prefer I4 matrix as it clearly distinguishes the objects when compared to the A4 matrix as A4 dithered image has uniform texture across entire image which makes the objects less clear to distinguish.
- For the improvement of the clustered dot screens we need a tradeoff between no of gray levels and resolution.
- While computing the image with 4 intensity levels i.e., 0,85,170 and 255 I have used thresholding to produce the image.
- That is if an image pixel lies in between 0 and 85 then if it is less than the average of 0 and 85 then the pixel is set to 0 otherwise 85.
- Similar procedure is followed for other ranges of intensities.

(b) Error Diffusion

2.5 Motivation:

- Error diffusion technique can be used to improve the quality of Digital Halftoning which is previously done by dithering technique.
- It produces such effect by diffusing error into the neighbouring pixels.
- By doing so we quantize the error and producing an effective digital halftoned image.
- This technique quantizes each pixel using a neighborhood operation, rather than a simple pointwise operation.

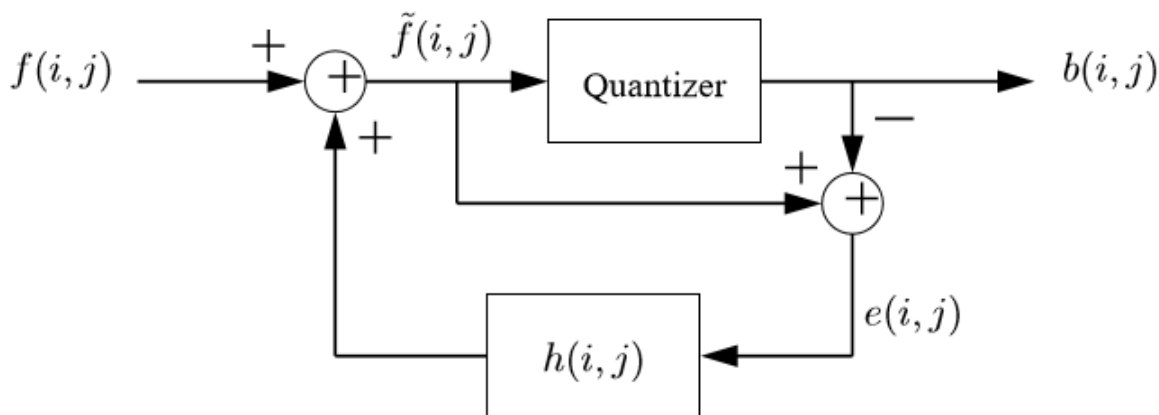
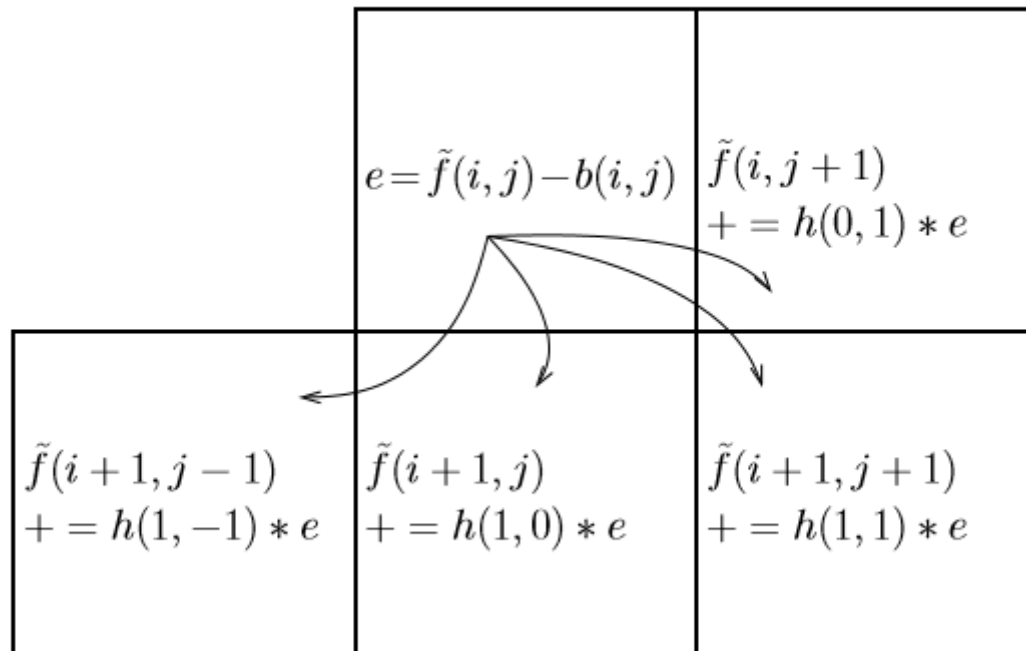


Figure 9 Filter View of Error Diffusion

2.6 Approach and Procedure:

- We have 3 different weight matrices to do error diffusion on, i.e., Floyd-Steinberg's error diffusion, Error diffusion proposed by Jarvis, Judice, and Ninke (JJN), Error diffusion proposed by Stucki.
- Normalize the inputs to the range of 0 to 1. Compare the values with the threshold, in my case it is 0.5.
- If the value is greater than 0.5 I set the value to 1 otherwise 0.
- Renormalise the values to pixel intensity values. Set this value as new pixel.

- Error can be calculated by the difference of old pixel and new pixel.
- The error obtained in this pixel is pushed towards its neighbouring pixels and then the above steps are repeated.
- Its depiction can be shown below,



- Floyd-Steinberg's matrix is given by,

$$\frac{1}{16} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{pmatrix}$$

- The error obtained in the center pixel is pushed towards its neighbouring pixels and it quantizes the error.
- Similarly, Error diffusion proposed by Jarvis, Judice, and Ninke (JJN) matrix is given by,

$$\frac{1}{48} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

- The same procedure is applied, instead more number of pixels around it get diffused by the error produced by the center pixel.
- And, Error diffusion proposed by Stucki matrix is given by,

$$\frac{1}{42} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

- Similar to that of JJN, but slight change in the weights of the matrix.

2.7 Results:



Figure 10 Error Diffusion by FS



Figure 11 Error Diffusion by JJN



Figure 12 Error Diffusion by Stucki

2.8 Discussion:

- Error diffusion is a better digital Halftoning technique than dithering. We can clearly observe that in our results.
- When we compare the three outputs obtained by error diffusion matrices, the one obtained by FS is not desirable.
- As the window size of FS is smaller when compare to the other two, the error obtained in the center pixel shows its immediate effect on the neighboring pixel.
- Whereas in the other two cases the error produced is spread over more number of neighboring pixels, reducing the effect of error.
- As we can see that the error is pushed towards the neighboring pixels in this case, we can consider a case where the error is pulled towards it from the neighboring pixels.
- It is just the reverse implementation of the pushing algorithm.
- The error from the neighboring pixels is multiplied with the corresponding weights and added to that of the center pixel.



Figure 13 Improved technique

- I have tried playing with the weights of the matrix using the forward pull of the errors method. I haven't achieved any evident improvement.

Problem 3: Morphological Processing

Shrinking, Thinning and Skeletonizing:

3.1 Motivation:

- Morphological processing is a popular image processing technique, which modifies images based on shapes.
- In this part of the assignment I have performed Shrinking, Thinning and Skeletonizing on the given images.
- Morphological processing is often used in object recognition to identify the objects and perform image manipulations on it.

3.2 Approach and Procedure:

- I have a grayscale image, which needs to be converted into binary image.
- Create an M array with the same dimension as the binary image.
- Create a search window, let $x(i,j)$ be the center ,if it is 0 set $M(I,j)=0$ otherwise calculate the bound.

X0	X1	X2
X7	X(i,j)	X3
X6	X5	X4

$$\text{Bound}=2*(X1+X3+X5+X7) + 1*(X0+X2+X4+X6)$$

- If the bound is 0 or 12, set $M(i,j)$ as 0.
- If the bound is not equal to either 0 or 12 form a string of $X0X1X2X3X4X5X6X7$ and compare with the conditional mask pattern table.
- If there is a match then set $M(i,j)$ as 1 and if it there is no match set $M(I,j)$ as 0 and copy the input to output.
- Now take the M matrix form a string $M0M1M2M3M4M5M6M7$, if $M(I,j)$ is 1 then compare the above string to the second pattern table.
- If there is no match with the second pattern table, erase $F(I,j)$.
- If there is a match with the second pattern table as well, then copy the input to the output file.
- Repeat these steps until desired output is obtained.
- The pattern tables for shrinking thinning and skeletonizing are a little different.
- I have performed 200 iterations to achieve the below results.
- To find the number of squares, I have traversed through the shrunked image, if the pixel is white then I increment the count by 1.

Number of Squares = 24

- Coming to the frequency of squares of different sizes, I have taken a 3*3 search window, where center pixel is $x(I,j)$.
- If I find a patch where $x(I,j), x(I,j+1), x(i+1,j)$ and $x(i+1,j+1)$ is white and the others black, I start counting until I find a black pixel.
- I keep in track of the columns I have been searching and keep in mind that I won't count if it exists between the same columns.
- I keep in track of the count, which I consider as the size of the matrix.
- I have copied the values of the sizes and generated histogram using MATLAB.
- The results are given below.

3.3 Results:

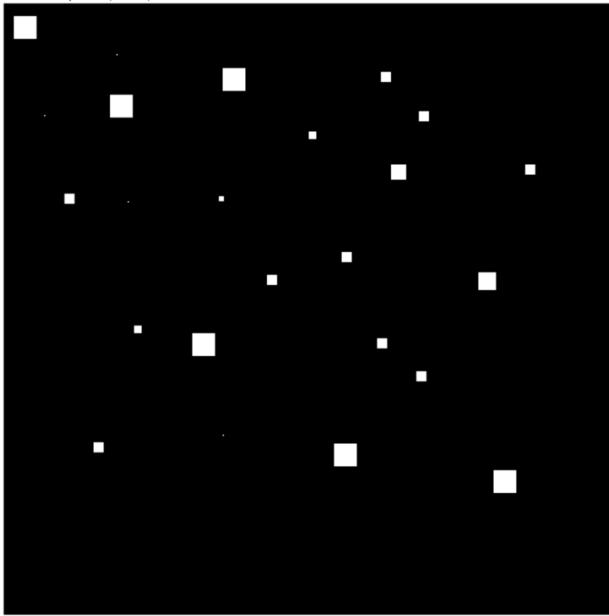


Figure 1 Original Squares Image

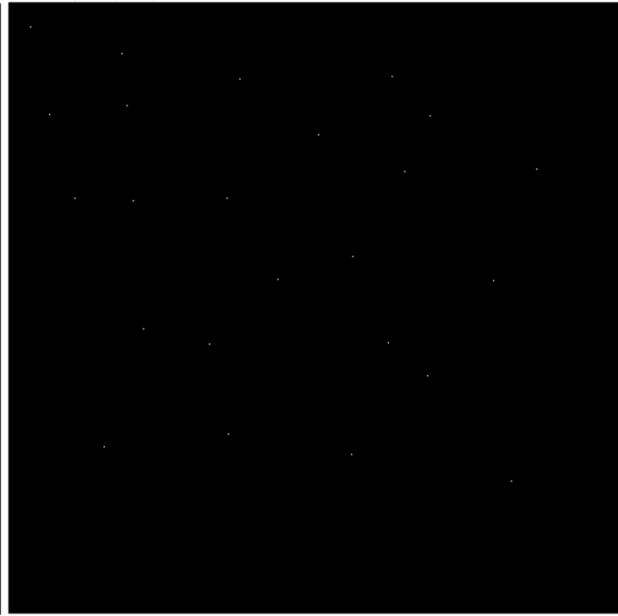


Figure 2 Shrunked image

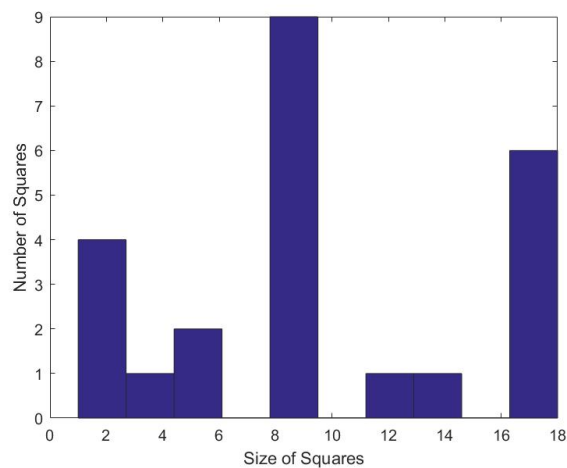


Figure 3 Histogram of Squares

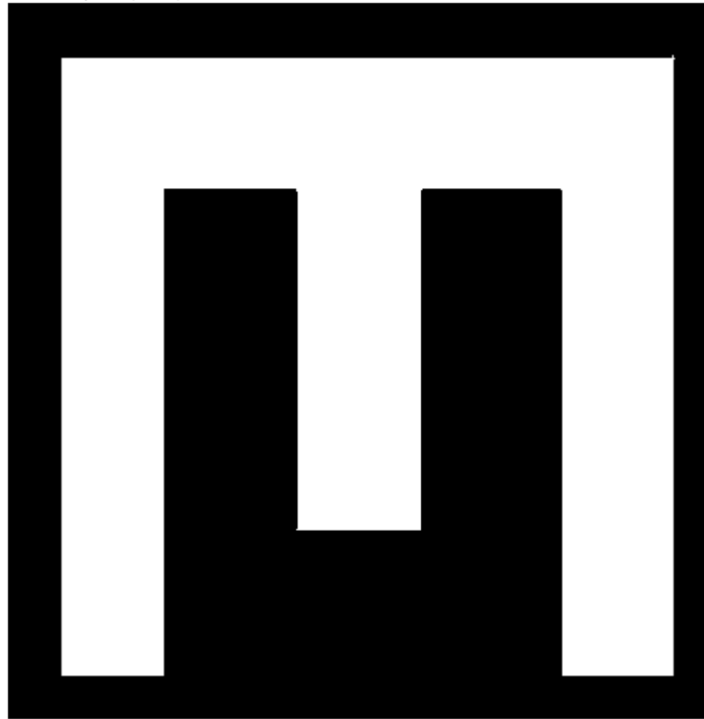


Figure 4 Original Letter E

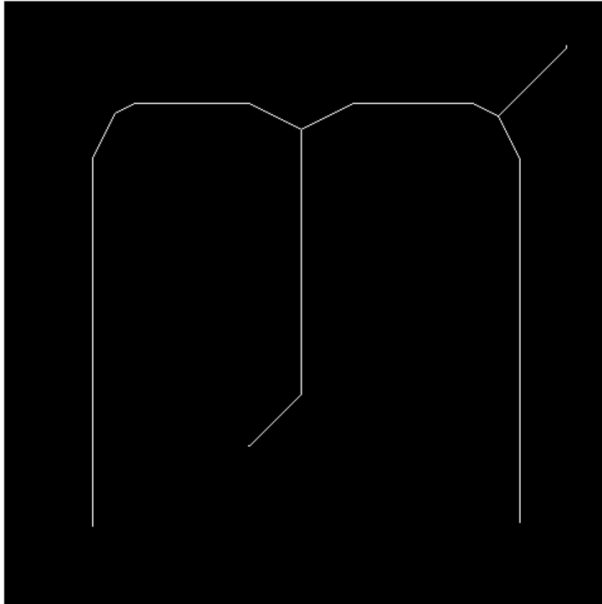


Figure 5 Thinning Output

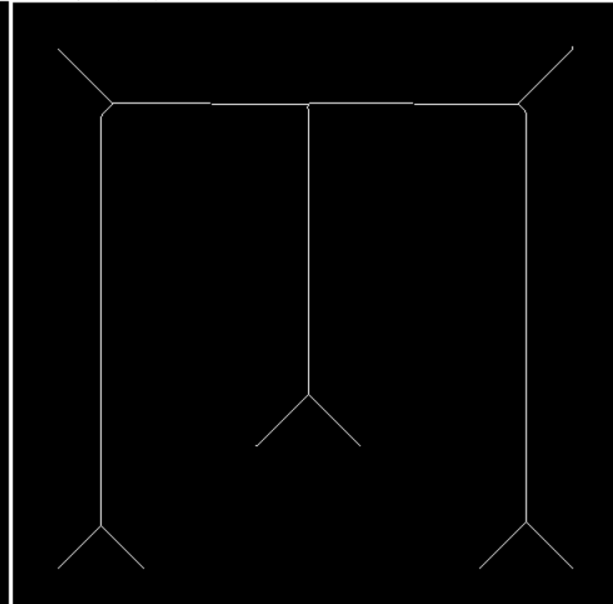


Figure 6 Skeletonizing output

3.4 Discussion

- Few challenges when I tried to do this problem were the generation of the pattern tables for each corresponding operation.

- The number of iterations to be made to achieve the desired output needs to be in mind, if not done the output may not be as desired.
- There is a slight discontinuity in the Skeletonizing of the letter E because of the shape of the line connecting to the other such rectangles.

Counting game

3.5 Motivation

- The above problem asks us to count various objects in the given board image.
- This technique can be used for object recognition in various image processing applications.
- Previous methods can be used to find the number of objects in the board game.

3.6 Approach and Procedure:

- First in order to find the number of white objects, we need to flood fill the holes present in it.
- Initially, invert the board image and apply the shrinking method that we have used in the previous problem.
- Then we get the number of holes and their center pixel position.
- Using this position try the flood filling algorithm where it says that it checks for the neighboring pixels, if it does not have the target color it starts filling until it finds the target color.
- Once the flood filling is done we get the white objects.
- Once again perform the shrinking operation, now we get the number of white objects.
- We need to find the coordinates of the centers of the white objects.
- Starting at the center try to move to right until we reach a black pixel, and note the pixel.
- Similarly find the length in all directions.
- Now a square has equal length on all sides, so increment the count if it satisfies the above condition, otherwise count it as a circle.

3.7 Results:

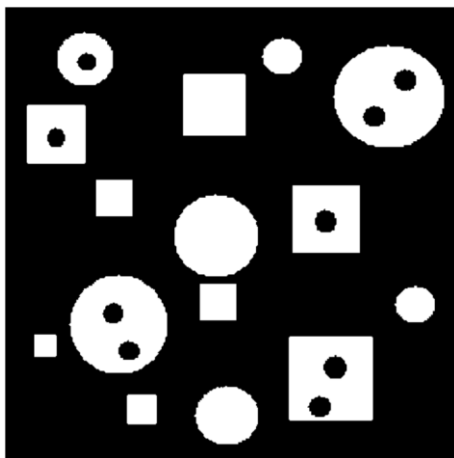


Figure 7 Original Board Image

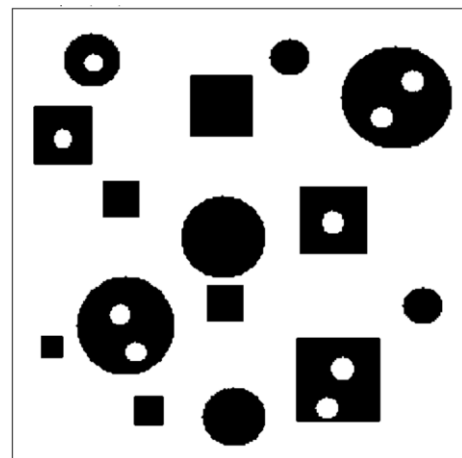


Figure 8 Inverted Image



Figure 9 Number of Holes

```
Number of Holes:9  
Coordinates for Flood Fill  
Row:56, Column:86  
Row:76, Column:426  
Row:115, Column:393  
Row:136, Column:53  
Row:225, Column:341  
Row:325, Column:114  
Row:365, Column:131  
Row:382, Column:351  
Row:424, Column:333
```

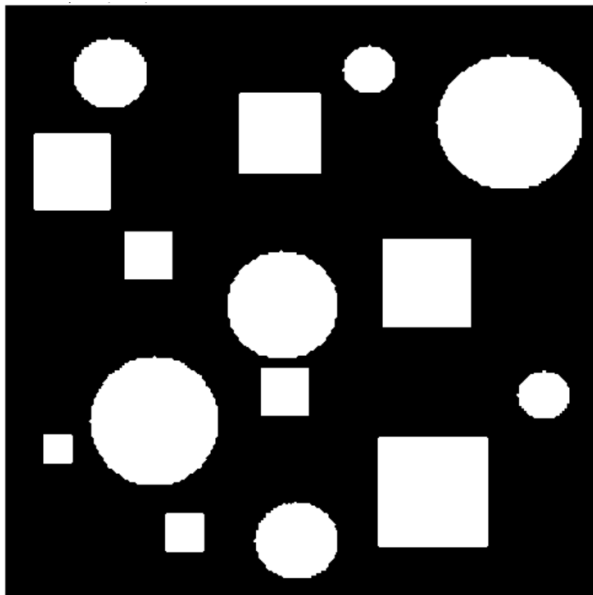


Figure 10 Image after Flood Filling

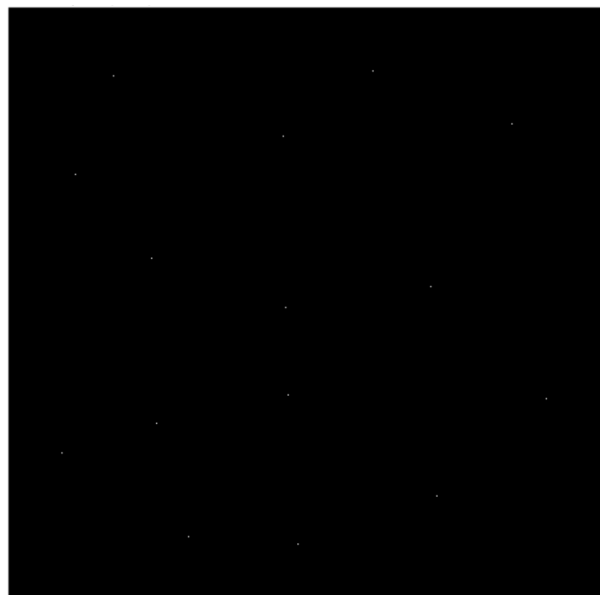


Figure 11 No of White Objects


```
Number of White Objects:15
Coordinates for White Objects
Row:50, Column:295
Row:54, Column:84
Row:93, Column:408
Row:103, Column:222
Row:134, Column:53
Row:202, Column:115
Row:225, Column:342
Row:242, Column:224
Row:313, Column:226
Row:316, Column:436
Row:336, Column:119
Row:360, Column:42
Row:395, Column:347
Row:428, Column:145
Row:434, Column:234
```

```
Number of Squares:8
Number of Circles:7
```

3.8 Discussion:

- Dilation and erosion could be used to count the objects, but due to the image constraints, it was very difficult to separate few objects based on the two techniques.
- Flood filling is a very effective alternative method to do so.
- Other challenging part was the unevenness of the shapes of the objects, which made it difficult to compare a specific patch to count the objects.
- Because of this unevenness I could exploit the size of the sides to find the different objects.