



DIGITAL IMAGE PROCESSING

EE569-HOMEWORK-2

SAHANA VENKATESH

6643244225(sahanave@usc.edu)




Table of Contents

Geometric Image modification	2
Motivation:	2
Approach and Procedure:.....	3
Geometrical Warping:	3
Puzzle Matching:	5
Image overlay and homographic transformation:	6
Results:	7
Discussion:	9
(2)Digital Halftoning	10
Motivation:	10
Approach and Procedure:.....	11
Dithering Matrix:	11
Error Diffusion Method:	11
Results:	12
Discussion:	14
Morphological Processing.....	16
Motivation:	16
Approach and Procedure:.....	16
Shrinking	18
Discussion:	20

G

eometric Image modification

Motivation:

The concepts of Image enhancement, Image modification and Warping are one of the building blocks in the field of computer graphics and image restoration. The idea behind warping is to change the location of the pixels without changing their intensity value. It can be viewed as a complement to the image enhancement technique seen earlier, where we change the intensities of the pixels. We will go through various techniques which make use of geometrical operations to modify shape of an image and have a hands on experience with their applications.

Ranging from the cups with the photo with the customized photo in shape of a diamond to the huge poster with a pizza photo in shape of a triangle, geometric image modification is seen in many things we are all familiar with. In essence, Image warping is about changing the spatial configuration of an image. We shall have a look at two different warping techniques and compare their results.

Hole filling is an interesting problem. Imagine having a Puzzle where you spotted a place where you want to place the piece in your hand but you notice that piece won't fit in if you don't modify it before placing it. When given sufficient information about the location of the holes and corresponding patches, we can utilize a hole filling algorithm which relies on the geometrical operations.

Merging two images by means of image compositing as seen earlier is easy. What requires more thought process and is wide spread is the idea of image overlay when we are dealing with two images taken from two different camera angles. Real world is in 3D co-ordinates and whereas the camera's image coordinate is two dimensional. So all the images taken by the camera is actually the projection of the world's co-ordinates into its 2D place. It is important to understand these equations so that when we try to visualize the object we can ignore the motion/angle at which the camera is placed while the picture is taken. We will learn about homographic matrix which helps us in the above step by taking into account the extrinsic and intrinsic parameters of the camera.

Approach and Procedure:

We need to convert Image coordinates to the Cartesian Coordinates because the operations such as rotation, translation become easier to imagine and work with in the Cartesian Coordinate system.

Let i and j denote the image row and column index.

$$x = j - (1/2);$$

$$y = h - i + (1/2) \text{ where } h \text{ is the height of the image.}$$

Geometrical Warping:

I Have implemented two techniques to solve the question.

Triangular Warping: The triangular warping method is about

1) finding a set of control points in the Input image and see where those points are mapped to in the output image.

2) Creating triangles bounded by these control points in both the input and output images in such a way that (1) all the points in the images lie in only one of these triangles and none of these triangles overlap. The triangular warping can be expressed mathematically as

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where x' and y' are Cartesian co-ordinates of the output image and x and y are the Cartesian coordinates of the input image and the matrix is the transformation matrix. I created eight triangles with control points (image corner points and centre points between two corner points).

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x'1 & x'2 & x'3 \\ y'1 & y'2 & y'3 \\ 1 & 1 & 1 \end{pmatrix} * pinv \begin{pmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ 1 & 1 & 1 \end{pmatrix}$$

here $(x'i)$ denotes the x coordinate of the control point of the triangle drawn in the output image and (xi) denotes the x coordinate of the control point of the triangle drawn in the input image. We calculate 8 such matrices. The mapping of a point in the input image to the output image depends on the triangle it lies in.

To avoid fractional output indices, we implement backward mapping where given the inverse of the transformation matrix and the output we are trying to find the input point which gave rise to the output in the first place. In case the input values are fractional, we use bilinear interpolation technique to find the intensity of the pixel at that location. The results obtained are discussed in the next section.

Linear Warping:

Linear warping is far simpler than the Triangular warping.

I divided the output image into two triangles (an imaginary horizontal line cuts the image into two halves).

Since we usually work with backward mapping, we start with the output image.

(1) Consider a point $u(i,j')$ in the output image. I assume diamond shape to be bounded by lines of slope 1. So the number of points at the height i is $(2*i+1)$. Here i and j are the image coordinates. We calculate scale factor as $(\text{number of points at height } i / \text{Width})$.

(2) you divide j' by the scale factor while keeping I constant. This gets mapped to j . We perform linear interpolation if j is fractional.

$$u(i, j') \rightarrow u(i, j).$$

So The points at the arbitrary height h gets mapped to the points in the output image at the height h (differing by their column index).

Puzzle Matching:

There are three steps involved in solving this problem.

The workflow is finding the position of the missing piece in the piece.raw image. Then finding the position of the points where I would like to place the piece. Then I would like a Warping function to place the piece in the missing place where it belongs.

1. I divided the piece image into two planes. I assumed that the Hillary's (image missing piece) coordinates would lie on the plane where $i < 260$. i , here refers to the image coordinate index of the piece image. Four points are selected in way mentioned below.

(i) point 1-all the pixels above the this pixel and to the left of the pixel is zero. There is a counter which turns 1 when such a point is encountered for the first time. This is to fix the point and to avoid overrides.

(ii) point 2-Store all the pixels for whom the pixels below to it and to the left of the it is zero. Find the pixel which has the smallest j coordinate among all the pixels encountered.

(iii) point 3-Store all the pixels for whom the pixels below to it and to the right of the it is zero. Find the pixel which has the largest i coordinate among all the pixels encountered.

(iv) point 4-Store all the pixels for whom the pixels below to it and to the right of the it is zero. Find the pixel which has the largest j coordinate among all the pixels encountered.

You repeat the steps to find out the coordinates of the trump image. The above method was implemented as intuitively make sense. If you look carefully at it, This simple algorithm is trying to find out the points where the change along i and j coordinate is maximum. This is simple and worked really effectively than the corner harris algorithm for me.

2. Then In the Hillary/Trump image you would scan through the image until you hit the first point whose r, g and b intensity is 255. Given the information that the holes are 100×100 . We find the other points using the above information.

3. We perform triangular warping. We divide the missing piece segment and hole segment into two triangles with their corners as control points and perform the triangular warping as discussed in the above section.

Image overlay and homographic transformation:

Here we are trying to combine two images which are assumed to be taken from two different camera perspective. First steps involve visiting the basic concepts which goes behind the 3D world coordinates to the 2D image coordinates of the camera.

$$w * \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f & 0 & Cx \\ 0 & f & Cy \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & tx \\ r_{21} & r_{22} & r_{23} & ty \\ r_{31} & r_{32} & r_{33} & tz \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where x,y and z are camera's cartesian co-ordinates. Here f is the focal length of the camera lens.

Here-refers to the matrix which takes into account the intrinsic features of the camera and **Here**-refers to the matrix which takes into account the extrinsic features of the camera.

In absence of the information about these features we have homographic transformation matrix which describes relation between two planes. Given two planes and given four points in one plane and the points where you would like them to be placed in another place, we can define a mapping. To find the elements of the homographic transformation matrix

$$\begin{vmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -y_3X_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -y_4X_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3Y_3 & -y_3Y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4Y_4 & -y_4Y_4 \end{vmatrix} \cdot \begin{vmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{vmatrix} = \begin{vmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{vmatrix} \quad [1]$$

Since we are interested in backward mapping X,Y refers to the Cartesian co-ordinates of the text image and x,y refers to the Cartesian coordinates of the field image.

The homographic transformation matrix is of the form $\begin{pmatrix} a & e & b \\ f & c & g \\ d & h & 1 \end{pmatrix} [1]$.

For all the pixels lying in between four points in the field image we use the homographic transformation matrix to find out intensity from their image's (each pixel in one image is assumed to be mapped to another pixel in text image, called the pixel's image) location in the text image. Since we are overlaying the text image in the field image, every pixel in the

text image(if not black in case of tartans.raw and not white incase of Trojans.raw) contributes to the half intensity at its image location and the other half of the intensity value is contributed by the pixels corresponding to the exact location form the field image.

Results:

1)Geometrical Warping:



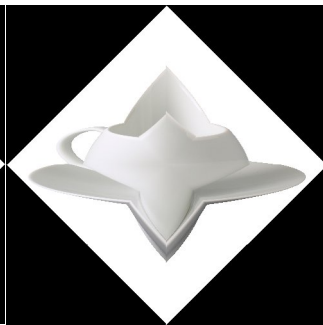
Input image



Output Given

by

Linear Warping



Output Given

by

Triangular Warping

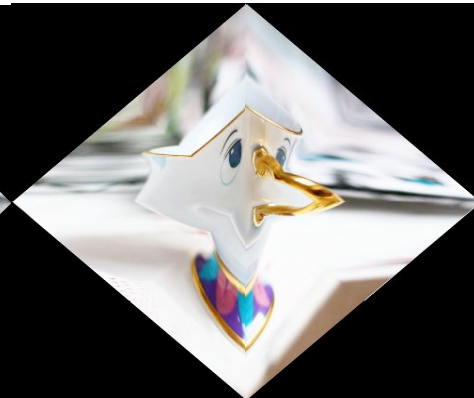


Input image



Output Given

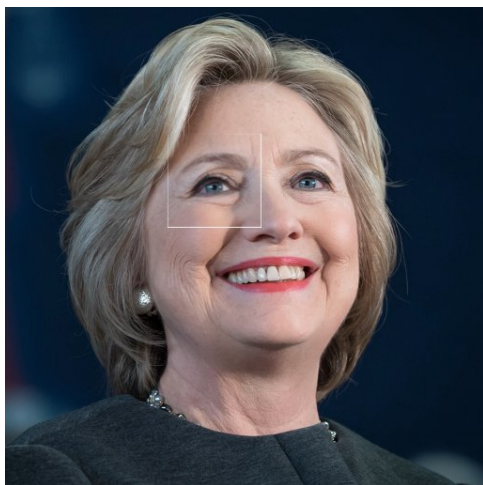
By Linear Warping



Output Given

By triangular Warping

2)Puzzle Matching:



Output of the Puzzle matching algorithm



Using mean filter to get rid of white gaps

Using Median filter to get rid of white gaps:



3)Output of the homographic transformation and Image overlay algorithm



Discussion:

Linear warping program implemented here would work if I want to warp any 500×500 image into a diamond shape image whose sides have slope of 1. So it needs a lot of modifications if you move out of this zone. The **Triangular warping** algorithm is more generalizable and it works well for multiple cases but fails to do a good job here. This is because the algorithm works on setting. Imagine you having multiple black boxes; you supply your point in one of the boxes (based on some threshold) and you obtain the intensity value. The key information seen here is in what triangle the point lies. We supply very little or less information about where the point lies in the triangle. The idea that pixels that are close to each other mostly have the same intensity values is lost here as we lost local information about pixels. Which in combination with the fact that the object is not symmetric has led to poor results in case of triangular warping. The triangular warping works very well

in cases where the object is shown to be symmetric .This is because symmetric object would imply triangles within the image to be similar to one another. This would cause introduction of edges in the output image and mapping almost similar for each triangle thus intuitively preserving the symmetry.

We have successfully used triangular warping *to fill the puzzle piece* in the main image. Close inspection showed colour variation between the puzzle piece placed in the hole and the main image .Viewing the white pixels which border the puzzle piece in the main image as a salt noise introduced by the hole filling algorithm filter ,gave the usual methods used to tackle them-Mean and median filter. I applied mean filter to the Hillary image and I could see the mean filter blurring out the important edges and over smoothing the image at the expense of removing the border and hence I used median filter for both Hillary and Trump images and It successfully removed the white border. Trump image as given out of the hole filling algorithm is blurry due to the poor interpolation.

Homographic transformations is performed by selecting the points in the field. Threshold is selected by the trial and error method. Since I was asked to do image overlay and not composting , I did not overwrite the original pixel information and I rather distributed the intensity of the pixel at the new location to be a function of both(text_image_pixel(controlled by a threshold),field image_pixel(at that location) and a ratio factor (between 0 and 1)).This process is easy to automated if we could specify the size of the text image and H matrix.

Digital Halftoning

Motivation:

Dithering helps us to manage when we have restricted colours .For example, In an image manipulation software when you convert an image to GIF format, it is restricted to 256 colours. In case of black and white image, one has to covert an image with grey scale values ranging from 0-255 to a cluster of binary values(255(1) or 0(0)).It is important to understand dithering because it makes us to concentrate only on performance of a printer a function of algorithm implemented rather than viewing performance of a printer as a function of its cost.

When you are quantizing something, you account for it mathematically by introducing a term called quantization error. The loss of bit resolution might make image to lose information. To account for it, Error diffusion is seen as a method to randomize the quantization error by introducing 'blue noise' to produce better results with printers with less colour.

Approach and Procedure:

Dithering Matrix:

Ordered dithering takes place with the help of Bayer's index matrices. Bayer's index Matrices are calculated using the formula shown in the homework. They are of sizes in powers of 2. We cannot have constant threshold because This would produce a poor quality rendering of a continuous tone image.

Algorithm:

- (1) Construct the N*N Bayer's index matrix. Here N is a power of 2.
- (2) Calculate Threshold matrix based on the Bayer's index matrices.
- (3) Loop the threshold matrix across the image.
- (4) This way there would be a threshold value assigned to every pixel and this would be value of the entry of the Threshold matrices above that pixel. The pixel is assigned 1 if its value is greater than the threshold value it is suppose to meet.

Different sizes of threshold maps exist:

$$\begin{aligned} & \frac{1}{4} \times \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \\ & \frac{1}{9} \times \begin{bmatrix} 0 & 7 & 3 \\ 6 & 5 & 2 \\ 4 & 1 & 8 \end{bmatrix} \\ & \frac{1}{16} \times \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix} \end{aligned} \quad \frac{1}{64} \times \begin{bmatrix} 0 & 48 & 12 & 60 & 3 & 51 & 15 & 63 \\ 32 & 16 & 44 & 28 & 35 & 19 & 47 & 31 \\ 8 & 56 & 4 & 52 & 11 & 59 & 7 & 55 \\ 40 & 24 & 36 & 20 & 43 & 27 & 39 & 23 \\ 2 & 50 & 14 & 62 & 1 & 49 & 13 & 61 \\ 34 & 18 & 46 & 30 & 33 & 17 & 45 & 29 \\ 10 & 58 & 6 & 54 & 9 & 57 & 5 & 53 \\ 42 & 26 & 38 & 22 & 41 & 25 & 37 & 21 \end{bmatrix}$$

[4]

We repeat steps (2),(3) and (4) for the A4 matrix.

The algorithm makes sense If we have 2 levels. In case we wish to express, our pixel intensities as one of N levels then we have to modify accordingly. The histogram of the man image revealed that mean of the pixel intensities are around 89 and so the threshold are [0,85,170,255]. As seen above it is really easy implementing Ordered Dithering algorithm.

Error Diffusion Method:

We wish to use common threshold and spread the quantisation error so that we don't observe blobs of colour or patterns. Error Diffusion Algorithms take a proportion of the quantization error and spread it across the neighbouring pixels. How far it spreads and the proportion differs between the algorithms.

Algorithm:

- (1) We are given the diffusion matrices of the Floyd, Jarvis and Stucki. We take one error diffusion algorithm at a time. We expand the image depending on window size of the error diffusion matrices.
- (2) Every pixel is binarized by keeping 0.5 (I normalized the pixel intensities) as threshold.
- (3) You spread the quantization error as per the error diffusion matrices.
- (4) Then you loop through the pixels in serpentine manner.

Results:

Input image:

Dithering using A(4) Matrix



Dithering using I(2) Matrix

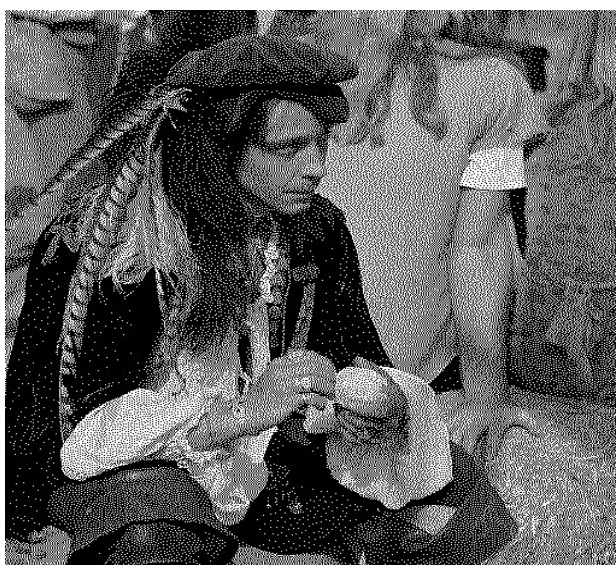
Dithering using I(4) Matrix





Floyd-Error Diffusion method

Error diffusion by Stucki method



Error Diffusion method proposed by Jarvis,Judice and Nicke



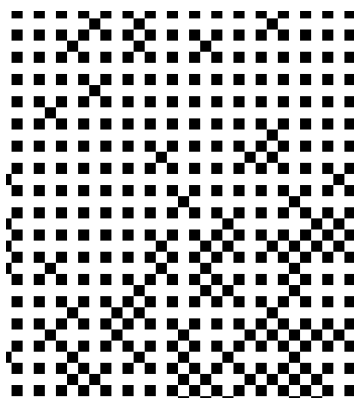
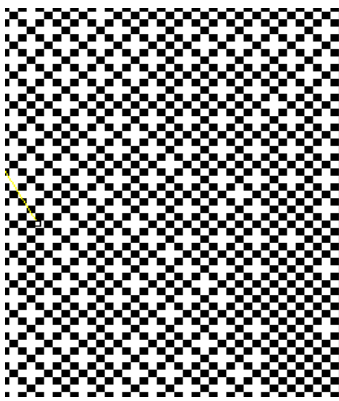
Discussion:

Upon close inspection we find that there are some patterns in the image after we pass it through the ordered thresholding matrices. Since there is a pattern behind applying threshold to the various pixels in the image this pattern gets translated into some sort of patterns in the output image. When you zoom in the image after applying thresholding by A4 matrix we see some kind of spiral pattern which implies A-4 applies some kind of spiral thresholding to the pixels.

Quantizing image to Four levels makes it look like it has enhanced the image. But application of the method has lead to loss of some of the edge information. These patterns are clearly visible in area where the pixels have the uniform intensity.

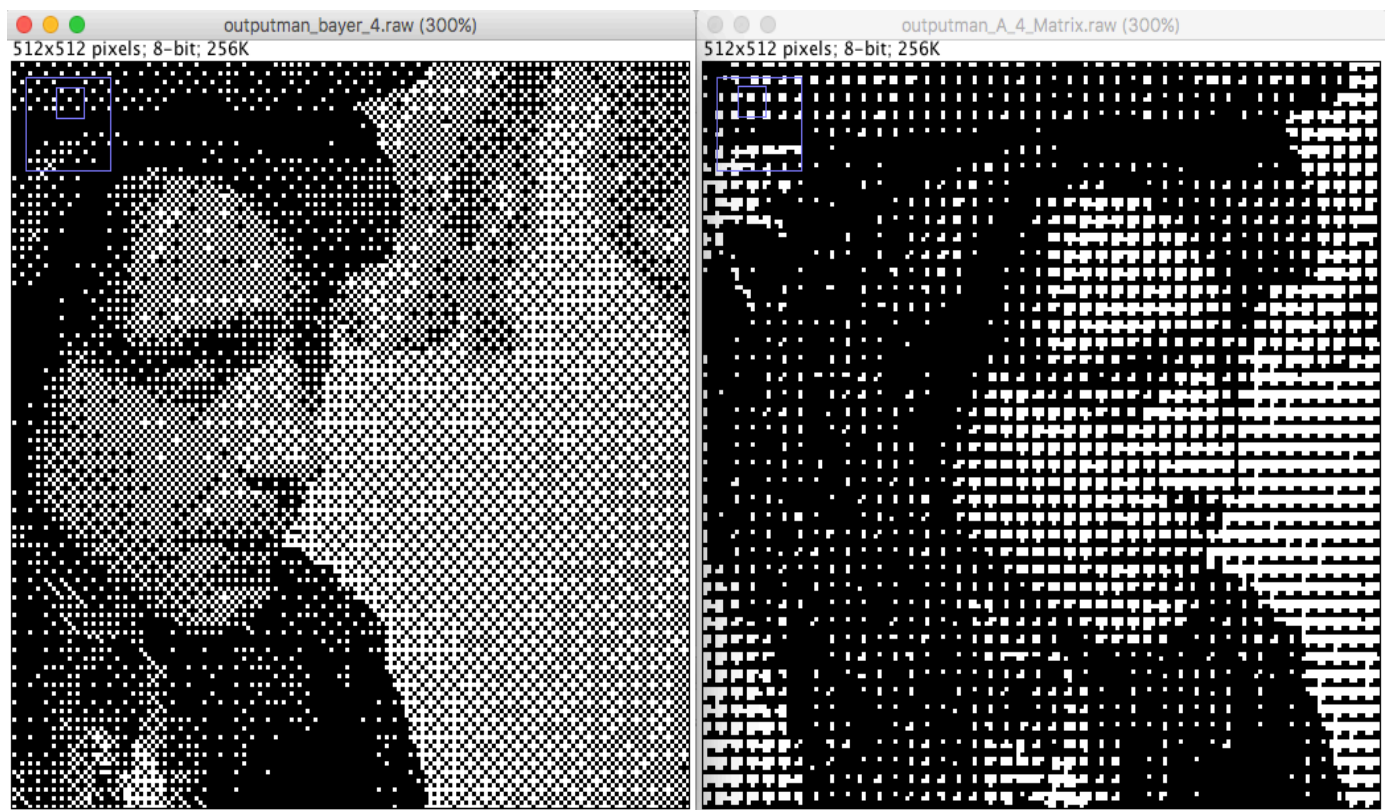
Patterns of: Bayer Index 1 8 matrix

Bayer Index 1 2 matrix



Patterns of: Bayer Index 14 matrix

A4 matrix



As seen from the above image, the patterns of A4 and I4 are different. Dithering by A4 has introduced new edges in the image and has preserved some of the old edges whereas the Dithering by I4 has clearly made the Image to lose some of the edges but It appears much more smoother and natural than the one given by A4.

Closer look at Output as given by Floyd Diffusion Algorithm:



We see that Performance of Error Diffusion by Jarvis algorithm and Error Diffusion by Stucki algorithm is much better than the error diffusion by Floyd because quantization error is spread to more pixels. In Floyd Diffusion algorithm, the error is spread to the immediate pixels which interferes with the smoothness of the image. There are so many distortions which can be viewed as high frequency noise because they were introduced by the Floyd diffusion algorithm.

Overall I find the performance of the Error diffusion algorithms to be better than Dithering matrices because of the quantization error which is spread around the neighbourhood of a pixel which makes sure the image does not have repetitive pattern. I find the performance of the Stucki and Jarvis to almost be the same .A closer investigation revealed that Algorithm by Stucki is better than Algorithm by Jarvis because Stucki matrix adds lesser error on the neighbouring pixels and there fore we suffer from lesser distortions than Jarvis(Have a look at the well defined nose in the image given by Stucki algorithm vs Distorted image given by Jarvis algorithm).



The only issue I find with its inability of the error diffusion algorithms to handle distortions. I propose a method which diffuses only a proportion of the error to its neighbours so that we suffer from lesser distortions. This is called reduced colour bleed.

Morphological Processing

Motivation:

Here in Morphological Image processing you don't care about the colour of the pixels you only pay attention to the shape/patterns in the images. In this Problem, we are trying to implement many operations with the help of hit/miss transformations. A Structural element', typically 3×3 , is scanned over a binary image. If the binary-valued pattern of the mask matches the state of the pixels under the mask (hit), an output pixel having same location as that of the location of the the centre pixel of the mask is set to some desired binary state. For a pattern mismatch (miss), the output pixel is set to the opposite binary state.

Approach and Procedure:

The algorithm to Implement Shrinking, Skeletoning and Thinning are similar and it happens in two stages as shown below.

(2) In the second stage of the algorithm, the centre pixel X and the conditional marks in a 3×3 neighborhood centered about X are examined to create an output pixel.

Algorithm:

(1) If(Image pixel !=0)->

Then check whether 3×3 window with the Image pixel in the middle of the 3×3 window with the table 14.3.1 (after calculating the bond) and check for patterns listed under the operation you wish to do by checking whether its S,K or T (*S for Shrinking and K for Skeletonizing and T for Thinning*) in the table column.

This is the first stage, the states of eight neighbouring pixels around a non zero Image pixel are gathered together along with the Image pixel and this is followed by a look up table. (14.3.2 for Shrinking and Thinning and 14.3.3 for Skeletonizing)

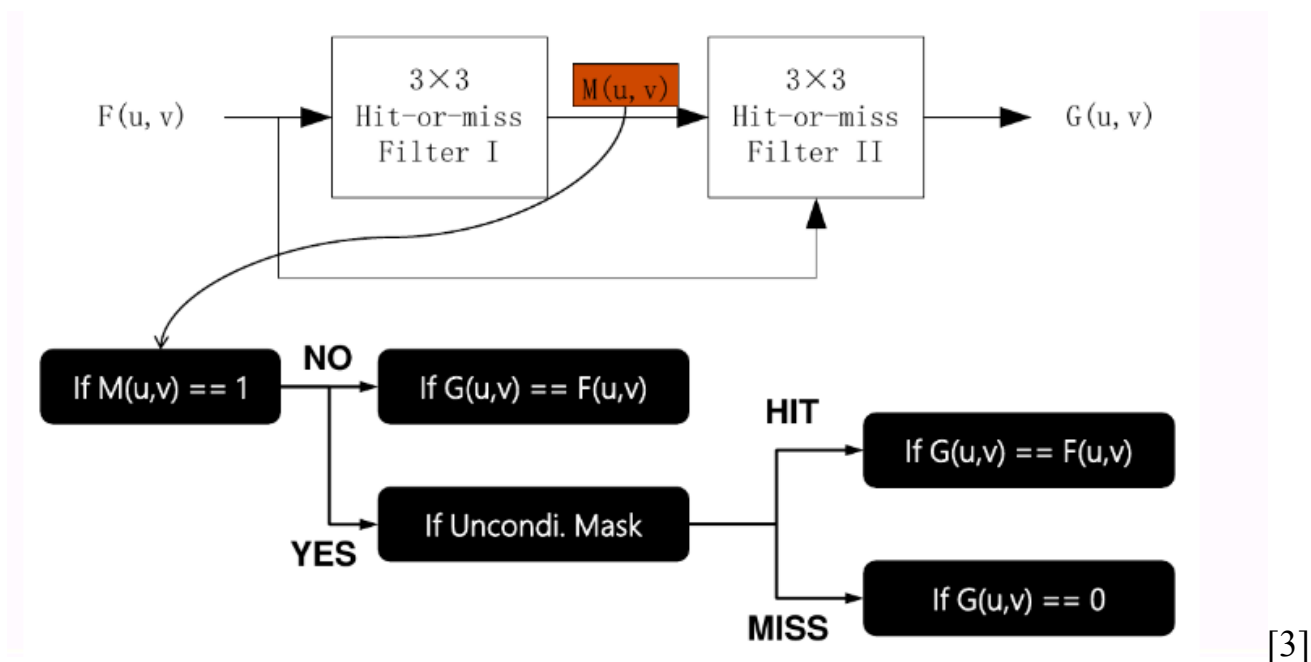
A look-up table generates a conditional mark M for possible erasures.

(2) Investigate the M value $M[i,j]$

After this process, you will have conditional Mask Matrix M which is of the size of the Input

image. If M is zero, then you copy the value of the input matrix to the output matrix.

Else, In the second stage of the algorithm, the centre pixel X and the conditional marks in a 3×3 neighbourhood centred about X are examined to create an output pixel.



Implementing the counting :

We implemented counting by using the following steps:

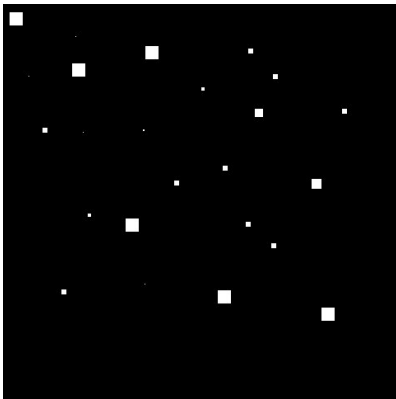
- 1) I complemented the image and performed erosion so that sides of the squares/circles after shrinking would not come close.
- 2) I performed Shrinking to get the white dots at the center of each square structure.
- 3) I counted the white dots which are surrounded by black pixels this gives me the number of holes in the image.
- 4) Then I complement the image again and run shrinking. The number of white dots will now give me the number of white objects in the image.
- 5) I will try to find a white pixel when I loop through the image and check whether it's the left most corner of a object. The adjacent corners are equidistant from a corner I run this logic and find the left most corner. I try to find the other two corners. If the distances are equal, it's a square otherwise it's a circle.

Results:

```
(Sahanas-MacBook-Pro:homework3 sahanavenkatesh$ ./shrink
total number of squares:
24
size i,number of squares of size i,
1,4
4,1
6,2
8,9
12,1
14,1
18,6
Sahanas-MacBook-Pro:homework3 sahanavenkatesh$ █
```

Shrinking

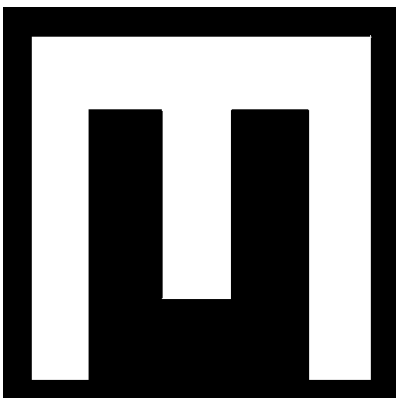
Shrinking after 1 operation



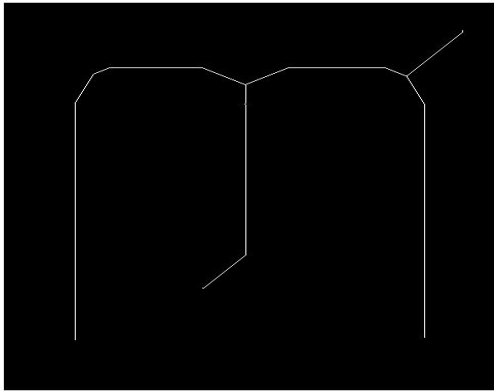
Shrinking after 9 operations



Input image

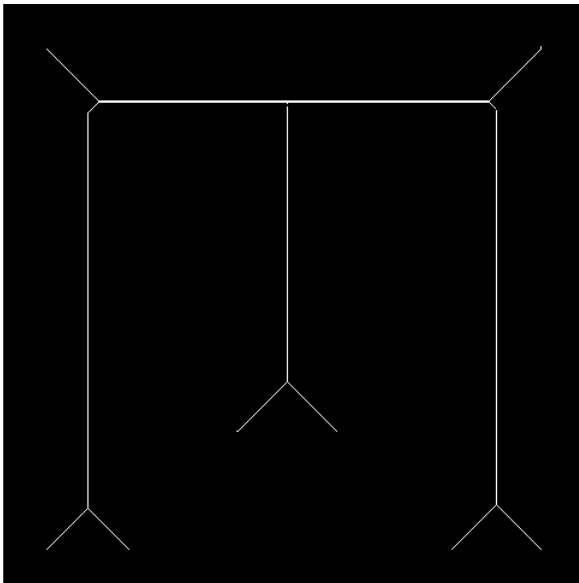


Thinning filter output:

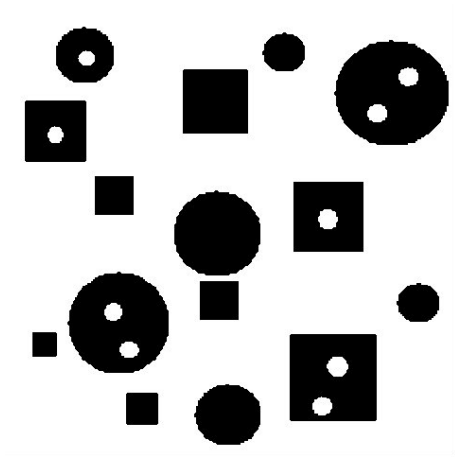


(Thinning after 70 iterations)

Skeletonizing filter(after 100 iterations)



After complementing



Discussion:

Shrinking: You erase pixels in such a way an object without holes erodes to a point at or near its centre of mass. A 3×3 -pixel object will be shrunk to a single pixel at its centre. A 2×2 -pixel object will be shrunk to a single pixel at its lower right corner. [2]. It is seen that once a square has shrunk, it is left undisturbed because it does not match any of the patterns in the look up table. This is the most important part of the algorithm. The Step one of the algorithm ensures that there is no complete erasure. The shrinking is idempotent, which means as soon after one point (number of iterations = 9), reapplications of the algorithm do not do any changes in the image.

When a square is shrunk it shrinks to the pixel closest to its centre of mass. Having this logic in mind, I calculated the number of white dots in the picture which came out to be 24.

I used the method used to find left most corners (from problem 2) and from corner I calculate the length by traversing through the border of the square. There is an array with 20 elements. Every time I encounter a square, array corresponding to that square gets incremented by 1. The different square sizes in the image and their frequencies are shown above.

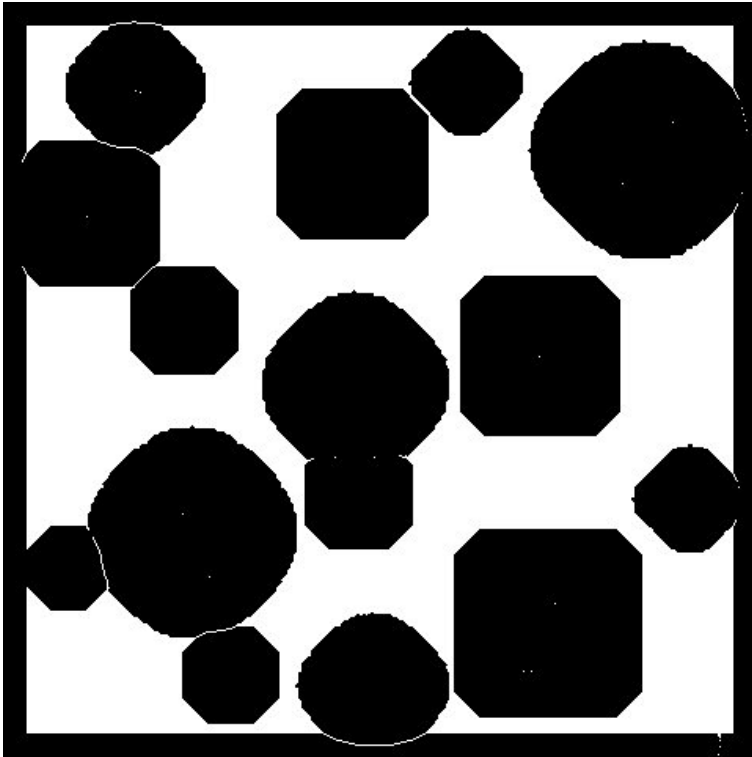
Thinning removes pixels from a set until only a narrow set remains. It is used to reveal set structure in recognition applications. In thinning the object would erode into a minimally connected point/line equidistant from its border. In thinning the would try to converge to a straight line that runs through its centre of mass. Complementing the image and then run thinning would help us to differentiate different objects.

*The need for **Skeletonizing*** rose when the thinning could not capture the skeleton/structure of the object. It is interesting how the skeletonizing works. It is keeping like keeping fire on E shaped structure and this is exactly how the fire would traverse. Skeleton captures the structure of the alphabet much more neatly.

Counting is an interesting application Of Shrinking.

Erosion so that sides of the squares/circles after shrinking would not come close. Because if the boundaries are distorted it would be hard to count the holes. For example,

Shrinking without erosion.Hence I used erosion.



Number of holes was found out to be 9.

Number of Squares was found out to be 8.

Number of circles is found out to be 7.

Number of White objects is found out to be 15.

References

- 1) http://www.cormap.com/features/homography_transformation.php
- 2) Digital image processing by prat
- 3) EE-569 Discussion-5
- 4) https://en.wikipedia.org/wiki/Ordered_dithering

