EE-569

INTRODUCTION TO DIGITAL IMAGE PROCESSING.

HOME WORK #2

Submitted by
Harikrishna Prabhu
3333077042
hprabhu@usc.edu

PROBLEM 1: GEOMETRIC IMAGE MODIFICATION

AIM:

To apply geometric modification and spatial warping techniques to do some of the interesting Image Processing tricks.

(a) Geometric Warping: Design and Implement a spatial warping technique to transform an input square image into an output disk-shaped image. As shown in the example below.

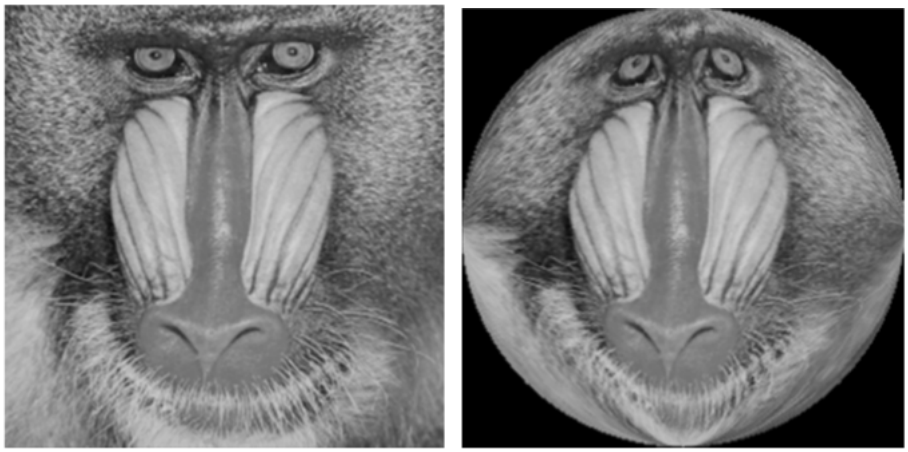


Figure 1: Warp the original Image (Square) to disk shaped.

Apply the same developed spatial warping algorithm to the given images in Figure 2.

Figure 2: Panda, Puppy and Tiger.

Then apply reverse spatial warping to each of the disk-shaped image to recover its original form. Compare the recovered image with the original image and if there is any difference? Explain the sources of distortions.

(b) Homographic Transformation and Image Stitching:

Implement the homographic transformation and stitching techniques to composite the room images shown below,

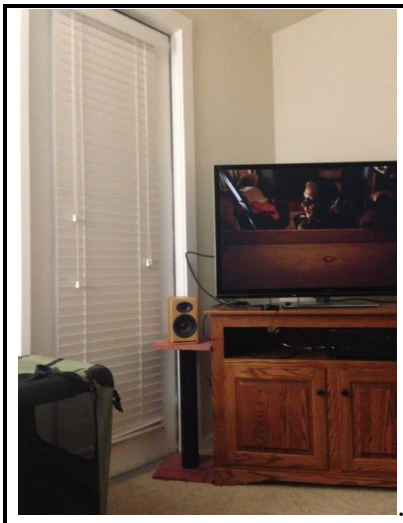
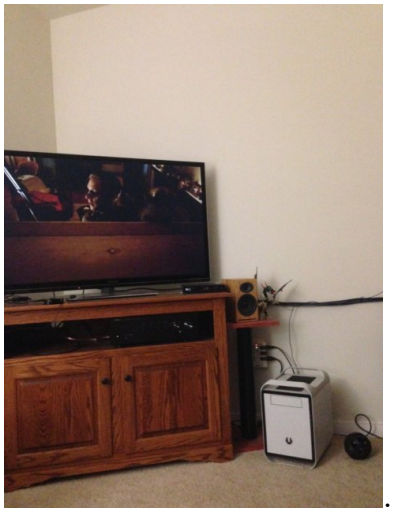
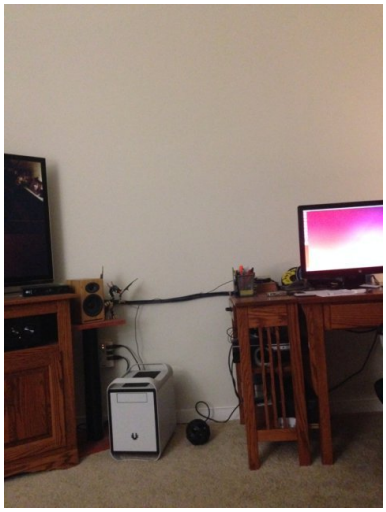




Figure: Left, Middle and Right.

Show the result and discuss the following question,

(i) How many control points were used? What if we use more that 4 control points.

(ii) How do you select your control points? Clearly state your observations and method used.

ABSTRACT AND MOTIVATION:

Geometrical Image Modification, in simple words, is a process of transforming the image coordinate system. That is, it deals with the spatial arrangement of the image data. It is often found to be important to perform geometrical modifications in cases such as, (i) misalignment of image that were taken with/at different sensors/time, (ii) Distortions brought about by the lens, (iii) Effects due to camera's orientation and other factors.

Image warping is a spatial transform process, where we digitally modify or manipulate the image data. In pure sense, warping means mapping an image point from one finite location in an image to another finite location in same/new image without modifying the color. This mapping can be done either in forward direction or backward (inverse mapping).

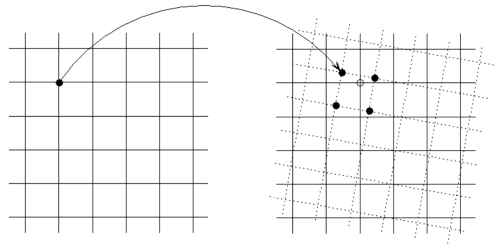


Figure3: Mapping

If the mapping co-odinates are not in finite space then interpolation is done.

Apart from pure warping, warping is generally done along with an effect. Majorly warping techniques is used in correcting distorted image or for creative purposes such as adding mirror effect, reflection effect along with scaling, translation and rotation.

Homographic Transformation: In Projective geometry, Homography is an isomorphism of projective spaces. This, Homographic Transformation is a very popular geo-referencing technique used worldwide. In simple words, Homographic Transformation is used to provide a relation between two images presenting the same object, but taken from different viewpoints. It is mathematically represented as, **P2=H x P2** where P1 and P2 denotes the images representing the common planar object and H is the homography matrix.

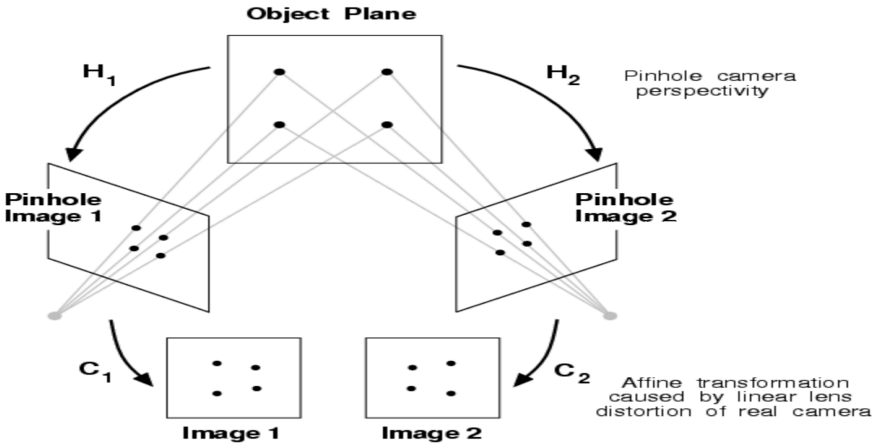


Figure 4: Homographic Transformation

This Homographic Transformation technique can be used to reconstruct a panoramic image with any number of photos that represent the common planar object.

APPROACH AND PROCEDURE:

(a) Geometrical Image warping:

The problem statement here is to warp the given square image to a disk-shaped image.

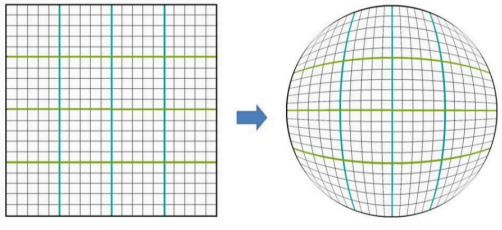


Figure 5: Square to Disc shaped mapping

Here while warping the image, the following conditions should be satisfied,

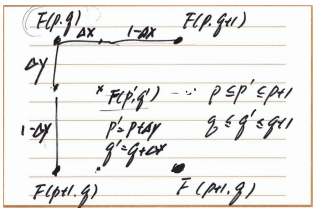
(i) Pixel at the centre of the square should lie in the centre of the disc as well.
(ii) Pixels that lie on the boundary of the square must lie on the boundary of the disc

Procedure:

- The original image is read as input image and an output image array is created of the same size of input as we won't be resizing the image.
- Coordinate transformation is done, that is from coordinates (x,y) of the original image we find the new coordinates (u,v) using the formula shown below.

$$u = x\sqrt{1 - \frac{y^2}{2}} \qquad v = y\sqrt{1 - \frac{x^2}{2}}$$

- Then to these new coordinates (u,v) of the output image, we copy the contents from the input image (x,y). It is not likely always that the coordinates produced using the formula will be a finite value. So, in such cases we can use bilinear interpolation technique.



- The points are forward mapped to the new output image. This resulting image will be the disc-shaped warped image.
- For reverse Spatial mapping

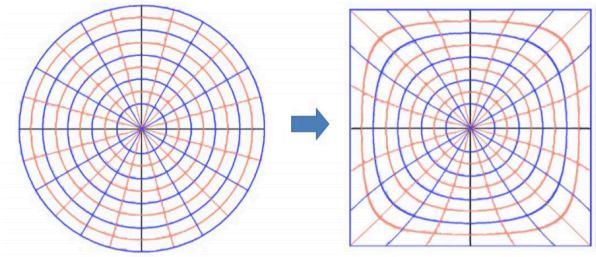


Figure 6: Disc to Square Mapping

we follow similar steps, where we find the new coordinates (x,y) from the disc-shaped image coordinates (u,v) using the formula,

$$x = \frac{1}{2}\sqrt{2 + u^2 - v^2 + 2\sqrt{2}\,u} - \frac{1}{2}\sqrt{2 + u^2 - v^2 - 2\sqrt{2}\,u}$$
$$y = \frac{1}{2}\sqrt{2 - u^2 + v^2 + 2\sqrt{2}\,v} - \frac{1}{2}\sqrt{2 - u^2 + v^2 - 2\sqrt{2}\,v}$$

Similarly, bilinear interpolation is done if necessary and the square shaped image is recovered from the disc-shaped image.

(b) Homographic Transformation and Image Stitching:

Here, we have to project the given three images (left, middle and right) into a single image. An example is shown below with 5 images being composite to one.

Figure 7: Homographic Transformation and Image Stitching example.

With how many number of images ever we use compose them to one. Still we follow this technique by picking images in pairs. So, the general rule for following with this technique is that, (i) Select control points from both the images (manual selection in also allowed), (ii)Apply homographic transformation to evaluate the homography mapping matrix. (iii) Using this transformation matrix, we warp one image over another.

$$\begin{bmatrix} x_2' \\ y_2' \\ w_2' \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad and \quad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x_2'}{w_2'} \\ \frac{y_2'}{w_2'} \end{bmatrix}$$

This is the equivalent to P2=H x P1.

Here, we see that the matrix has 3 dimensional variables. This is because we have augmented the 2-D space into 3-D. Therefore, the transformation matrix has 9 parameters and the parameter H_{33} is unity, hence it is sufficient to calculate 8 parameters.

NOTE: While adding images onto another, we make sure that we create an output image file large enough to hold all the input images.

Procedure:

- Take two images (P1 and P2, one image enlarged to hold entire set of input images). Choose 4 control points from both the images to compute 8 linear equations. This is done with ease using the cp.select feature in MATLAB.

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

Figure 8: H matrix parameters

Here, a=H11, b=H12 …….h=H32 and H33=1.

- Solve the 8 equations to deduce the transformation matrix H.
- In our problem, we follow inverse/reverse mapping. We follow $\mathbf{H^{-1} \times P2 = P1}$, that is for every coordinate with respect to P2, we compute the coordinates for P1 using the H inverse matrix.
- If that generated coordinates falls within the image's dimension range then we map, then we map one point over another respectively.
- Here first we apply this Homographic Transformation and Image stitching technique to left and middle image (middle image is the enlarged one to hold the left and the right image). Following steps 1, 2 and 3 we stitch Image Left to the Image Middle.

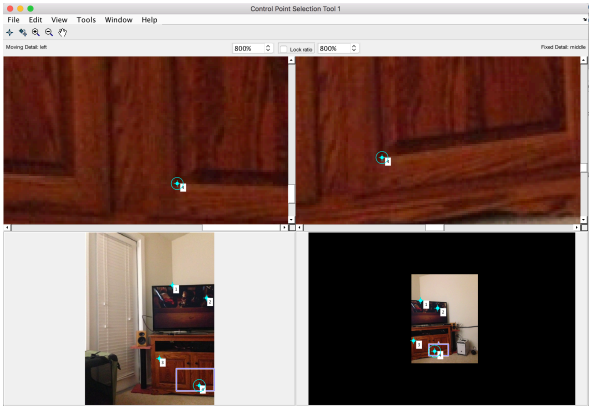


Figure 9: Left and Middle Image

- Once the Left is stitched to the Middle, we repeat the same steps to transform and stitch the Right onto the middle.

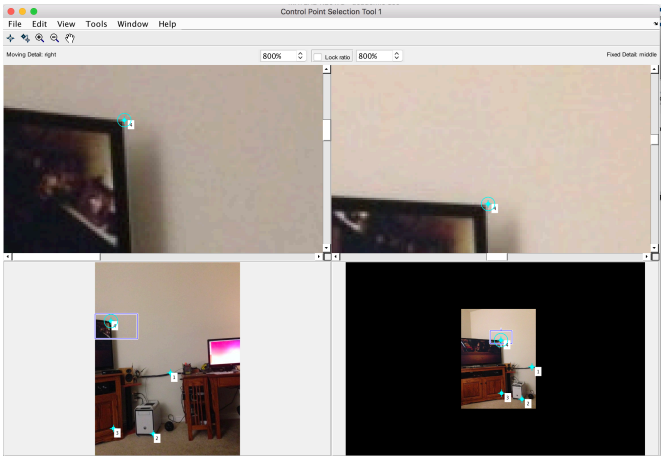


Figure 10: Middle and Right image.

- Since we manually select the control points, we tend to make errors very naturally and the final stitched image obtained is not much satisfactory.
- Hence using trial & error method, we select the control points and follow the above given steps until we get a desired output. That is, once we select apt/proper control points we can generate the stitched image as desired. Until then we might face difficulties in stitching the

new image that has error/miscalculations in its scale, rotation or translation value (That is, its projected image will not fit properly into the main image) .

EXPERIMENTAL RESULTS:

(a)Geometrical Warping:

Figure 11: Panda (Given)

Figure 12: Panda Disc shaped

Figure 13: Panda Square Reconstructed

Figure 14: Puppy Original

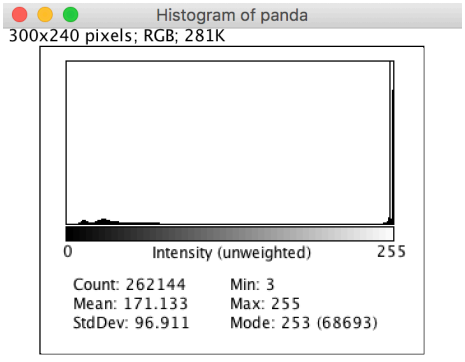
Figure 15: Panda Disc shaped

Figure 16: Panda Square Reconstructed

Figure 17: Tiger Original

Figure 18: Panda Disc shaped

Figure 19: Panda Square Reconstructed



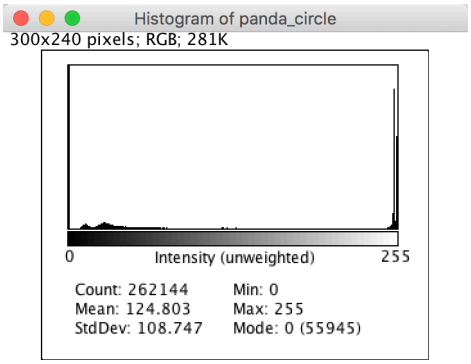


Figure 20: Histogram of Panda and Panda disc to show that there is no manipulation in its color components.

(b) Homographic Transformation and Image stitching:

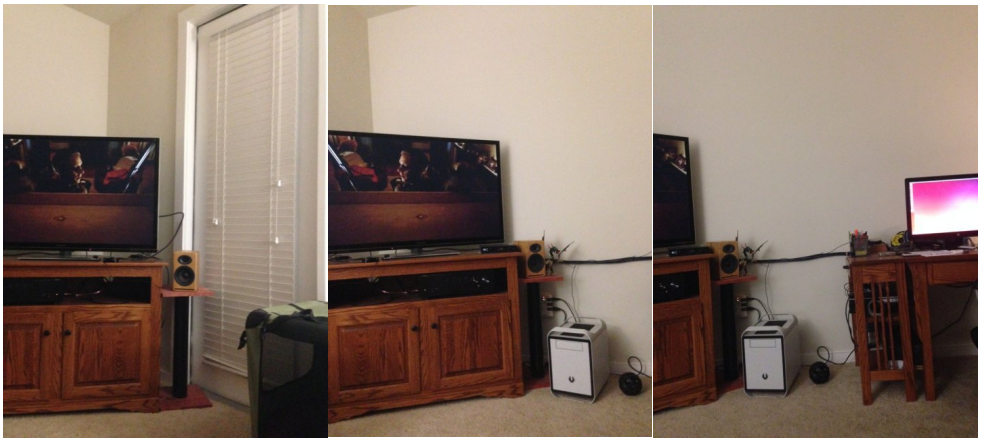
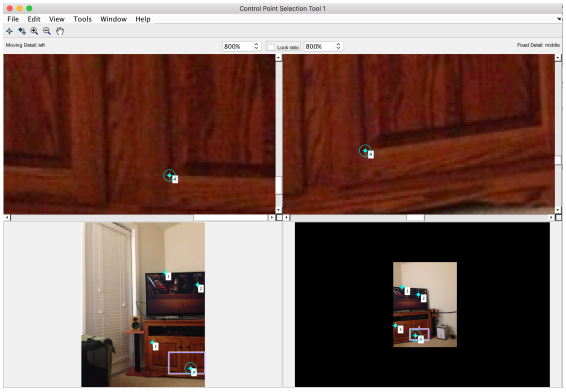


Figure 21: Left, Middle and Right



Editor – ctrl_pts.m

fixedPoints | movingPoints

4x2 double

	1	2
1	807.3125	489.6875
2	930.3125	544.4375
3	755.9375	777.9375
4	909.1875	853.1875
5		

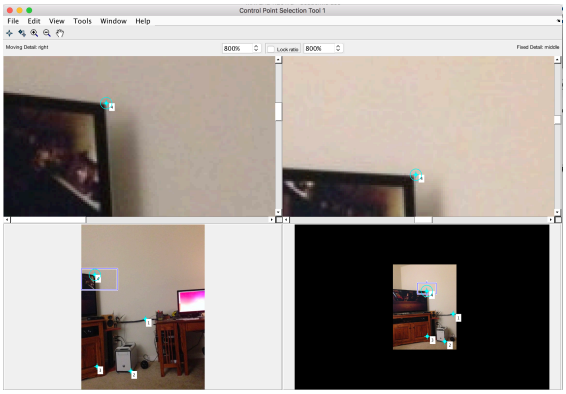
Editor – ctrl_pts.m

fixedPoints | movingPoints

4x2 double

	1	2	3
1	326.1875	195.9375	
2	452.4375	243.1875	
3	276.5625	468.0625	
4	425.4375	568.1875	
5			

Figure 22: Left and Middle Control Points and its coordinates.



Editor – ctrl_pts.m

fixedPoints | movingPoints

4x2 double

	1	2
1	1.1947e+03	672.6875
2	1.1307e+03	879.0625
3	1.0003e+03	840.0625
4	995.5625	501.1875
5		
6		

Editor – ctrl_pts.m

fixedPoints | movingPoi

4x2 double

	1	2
1	250.1875	366.8125
2	193.5625	568.8125
3	61.6875	550.9375
4	53.4375	195.8125
5		

Figure 23: Right and Middle Control Points and its coordinates.

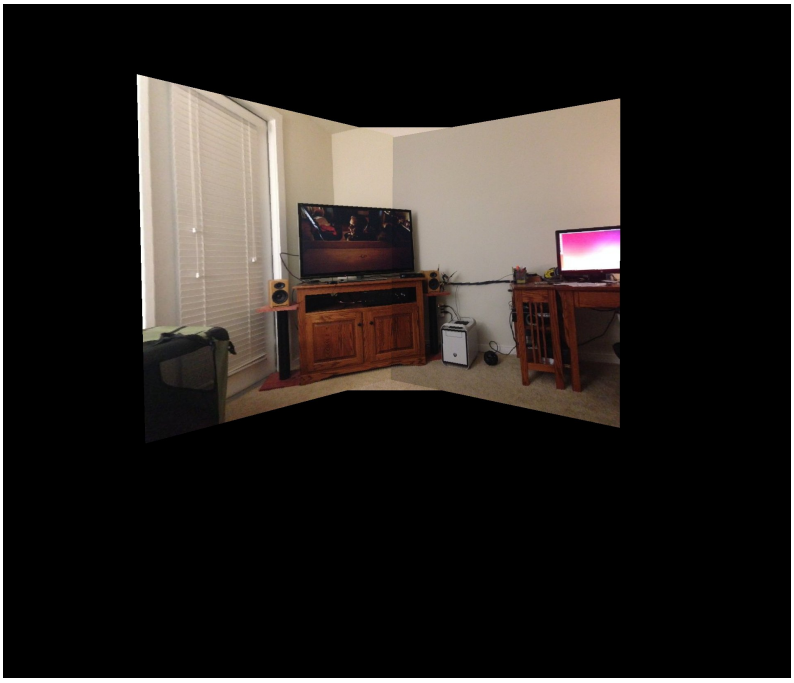


Figure 24: Final Stitched Image

DISCUSSION:

(a) Geometrical Warping:

It can be seen from the observation that the images: panda, puppy and tiger are warped into disc-shaped image and from the disck-shaped images, the square images are reconstructed successfully. And also that the conditions for warping is successfully followed.

This disc shaped warping can be obtained in many ways. Such as, (i) Normal square to disc stretching, (ii) Squircular Mapping, (iii) Elliptical grid mapping and so on. Here in this problem, I have used the elliptical grid mapping technique to get the desired output. The mapping is done starting from the major axis of the ellipse and proceeds with all possible ellipses that can be formed with this axis as the major axis around it.

NOTE: From the histogram of panda.raw and panda_circle.raw, we can see that during warping we haven't changed any of the color components of the image.

While comparing the original square image with the square-reconstructed image, we can see that there is mild distortions at the boundries of the image. This is because of the fact that, while mapping the original picture to the disk shaped image, we do bilinear interpolation (as not all finite coordinates of the square image fit perfectly onto the disc). During this process some information gets lost and while reconstructing the disc back to square, we get back those lost information as distortions in the final output image.

(b) Homograpic Transformation and Image stitching Technique:

480 x 640

1920 1640

1) In our problem, we use 4 control points. We use 4 control points because,

For our 2-D image homography we manipulate in augmented using 3-D, we need 9 parameters out of which one H33 = 1. So inorder to compute the other 8 parameters, we require 8 linear equations and that is obtained by choosing 4 control points in each image.

Incase we choose more than 4 control points,

(i) From the above explanation, we can see that for computing the H matrix, 4 control poins are equally sufficient(minimum requirement). Even if we take more than 4 control points,i.e more than 8 linearly independent equations to solve for 8 unknown parameters, we can deduce the H matrix.

Suppose we take more than 4 control points, we have more number of equations and same 8 unknowns parameters. So in that case we can compute the pseudo inverse to calculate the parameters of the H matrix. Not on the matrix we can match the control points in the image. So greater is the number of control points, better is the stitching output.

Fun fact: We can do homography for 3-D images, i.e. for homography for 3-D images, we'll be using 4-D augmented space for manipulation and hence the homography transform matrix will be of dimension 5x5. That is, we'll have to solve for 24 linear equations to compute the 24 parameters and the 25^{th} parameter H55 = 1.

$P2 = [H]_{5x5}$ x P1, where P1={x,y,z, 1, 1} and P2={X',Y',Z',const1,const2} and X=X' (combination of const 1 & const 2) and similarly for Y and Z.

2) How to choose control points:

The program used to choose control points and compute the H and H inverse matrix is shown below,

```
% MATLAB CODE for choosing the control points and computing the H inverse matrix:
left=imread('/Users/rickerish_nah/Documents/trial//matlab/left.png');
figure, imshow(left);
middle=imread('/Users/rickerish_nah/Documents/trial/matlab/middle.png');
right=imread('/Users/rickerish_nah/Documents/trial/matlab/right.png');
figure, imshow(middle);
hh=cpselect(right,middle);
% close(h);

x1=[fixedPoints(1,1) fixedPoints(1,2);
    fixedPoints(2,1) fixedPoints(2,2);
    fixedPoints(3,1) fixedPoints(3,2);
    fixedPoints(4,1) fixedPoints(4,2)];

x=[movingPoints(1,1) movingPoints(1,2);
    movingPoints(2,1) movingPoints(2,2);
    movingPoints(3,1) movingPoints(3,2);
    movingPoints(4,1) movingPoints(4,2)];


p=[x(1,1) x(1,2) 1 0 0 0 (-1*(x(1,1)*x1(1,1))) (-1*(x(1,2)*x1(1,1)));
   x(2,1) x(2,2) 1 0 0 0 (-1*(x(2,1)*x1(2,1))) (-1*(x(2,2)*x1(2,1)));
   x(3,1) x(3,2) 1 0 0 0 (-1*(x(3,1)*x1(3,1))) (-1*(x(3,2)*x1(3,1)));
   x(4,1) x(4,2) 1 0 0 0 (-1*(x(4,1)*x1(4,1))) (-1*(x(4,2)*x1(4,1)));
   0 0 0 x(1,1) x(1,2) 1 (-1*(x(1,1)*x1(1,2))) (-1*(x(1,2)*x1(1,2)));
   0 0 0 x(2,1) x(2,2) 1 (-1*(x(2,1)*x1(2,2))) (-1*(x(2,2)*x1(2,2)));
   0 0 0 x(3,1) x(3,2) 1 (-1*(x(3,1)*x1(3,2))) (-1*(x(3,2)*x1(3,2)));
   0 0 0 x(4,1) x(4,2) 1 (-1*(x(4,1)*x1(4,2))) (-1*(x(4,2)*x1(4,2)))];


b=[x1(1,1);x1(2,1);x1(3,1);x1(4,1);x1(1,2);x1(2,2);x1(3,2);x1(4,2)];

H=p\b;
H(9)=1;
m(1,1)=H(1);
m(1,2)=H(2);
m(1,3)=H(3);
m(2,1)=H(4);
m(2,2)=H(5);
m(2,3)=H(6);
m(3,1)=H(7);
m(3,2)=H(8);
m(3,3)=H(9);
h=inv(m);


y= ((h(1,1)*x1(1,1))+(h(1,2)*x1(1,2))+(h(1,3)))/(h(3,1)*x1(1,1))+(h(3,2)*x1(1,2))+(h(3,3));
```

Figure 25: Code to extract Control points and evaluate H matrix.

Here, I use a command called cp.select(moving image, fixed image). Where in case (1) Moving Image is Left.png and in Case 2 its Right.png. In both the cases we keep the Middle.png as fixed. Case 1 is shown below.

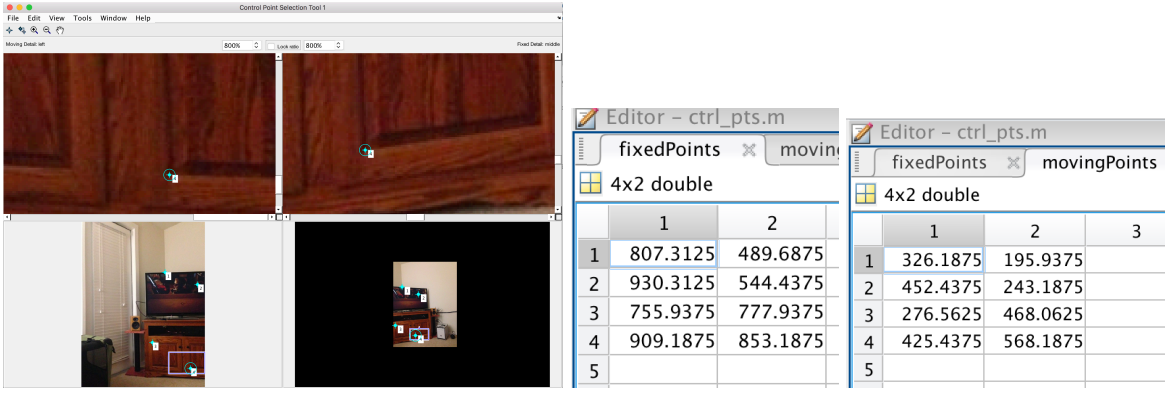


Figure 26: Left and Middle and its 4 control points.

This command helps us to extract the coordinates from the two images as shown (left- moving points and middle-fixed points). Then these coordinates are used to generate the 8 linear equations and they are solved to deduce the H matrix.

PROBLEM 2: DIGITAL HALF-TONING

AIM:

(a) Dithering:

Implement the following four methods to convert the given image, colorchecker, to half-toned images.

(i) Fixed Thresholding

(ii) Random Thresholding

(iii) Thresholding using Dithering matrix

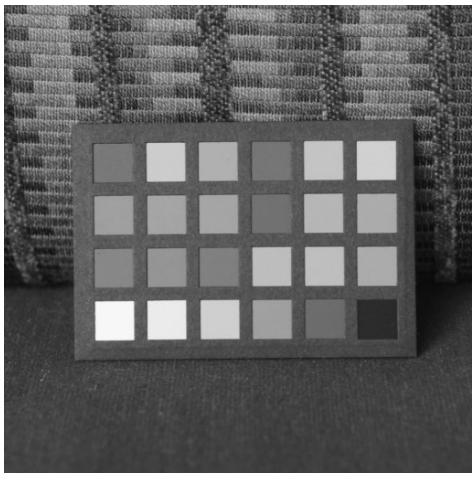


Figure 27:colorchecker

(1) Create I_2, I_4 and I_8 thresholding matrices to half-tone the given image.

(2) Create a display ready colorchecker image that has 4 grey-levels $(0, 85, 170, 255)$ and to explain the design idea and the algorithm.

(b) Error Diffusion:

To convert the colorchecker.raw image to a halftoned image using error diffusion techniques. A few are,

(i) Floyd-Steinberg method of error diffusion (Use Serpentine scanning).

(ii) Jarvis, Judice and Ninke (JJN) method of error diffusion.

(iii) Stucki method of error diffusion.

To compare the outputs produced using different techniques, mention the preffered method and to justify the method preffered.

The error diffusion matrices are,

Floyd-Steinberg error diffusion matrix: . Jarvis, Judice and Ninke (JJN) error diffusion matric:

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

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

Stucki method for error diffusion:

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

(c) Color halftoning with error diffusion:

(i) Seperable Error Diffusion:

To achieve color-halftoning for the given color image. Show and discuss the result of color halftoning and describe the shortcomings of this method.

(ii) MBVQ based Error diffusion: Implement the MBVQ algorithm to the given color image (Flower) and diplay the output image. Compare its results to that of the previous one.

Figure 28: Flower

ABSTRACT AND MOTIVATION:

HalfTone or halftone image is an image compromised of discrete dots instead of continuous tones. When viewed from distance, these discrete dots produce an illusion of continuous tones(lines or shapes). The main aim of halftone is to make printing inages more economic. That is, amount of ink used to print conituous tones is more than that for printing dots. Halftoning find its major application in all prining related fields. Eg, Newspaper, Magazing etc.

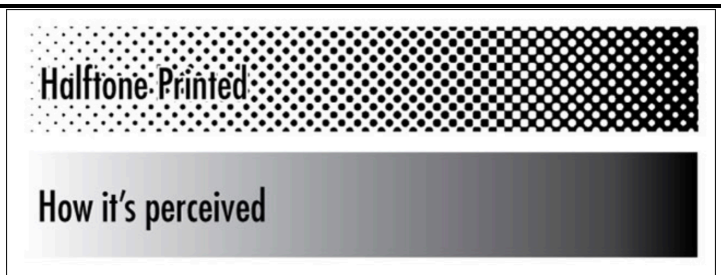


Figure 29: Halftoning Intuition.

(a) Thresholding/Dithering: Dithering in simple terms means to diffuse error at random to an image. Using Dithering technique in an halftoned image, we can reproduce tones of unavailable colors by mixing and matching pixel arrangement and intensity. There will be times where we will have only limited colors and we'd have to reproduce different variation in colors (Web designing, Multimedia Application etc.), there we can re-arrange the available color/tones to closely reproduce the desired unavailable color/tone.

Thresholding is a process of converting a greyscale image into a binary image,i.e the image is made up of only black and white pixels (0 or 1 [standardized]).

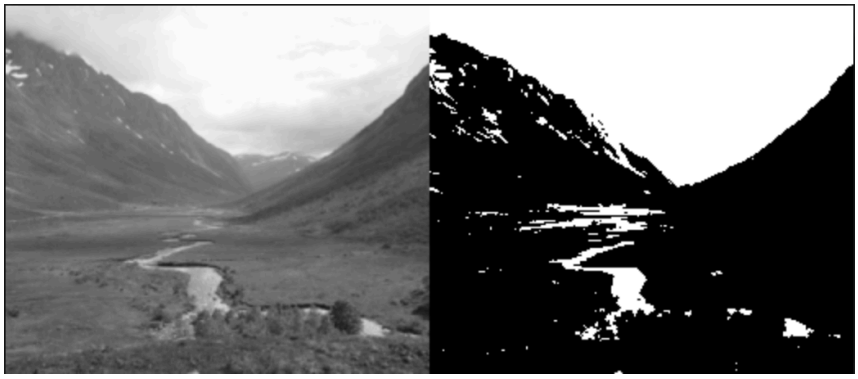


Figure 30: Greyscale image -> Threshold Image

There are so many ways to dither and threshold an image so that we can make it printer-friendly.These methods are very much helpful in the sectors with display and printing applications. A few are, (i)Fixed thresholding, (2)Random Thresholding, (3) Thresholding using dithering matrix.

(b)Error diffusion techniques for halftoning the images:

Error diffusion is also a method to halftone images. In previous dithering cases, we performed the halftoning process pixelwise where an effect to a pixel only affects that particular pixel. But here, an effect applied to a pixel not only affect that pixel but also to all its neighbouring pixels. The radius of affect depend on the window sizes preffered for each method. Error diffusion can be performed in both 1-D and 2-D. Using 1-D error diffusion keeps pushing the dots in one direction only and produces a distracting image. Meanwhile in 2-D error diffusion, we equally spread the error. In 2-D error diffusion one has to make a note that as we proceed further with equally redistributing the error, we must not disturb the pixel that has already been processed.

1) Floyd-Steinberg error diffusion method: It is arguably one of the most saught after and simple error diffusion techniques. It is very simple and effective to implement.
2) Jarvis, Jude and Ninke error diffusion method: This method is quite stronger and quality efficiemt model, but it is not quite popular because of its complexity in implementation. In comparison with Floyd-Stienberg's method , we can say that, here the error diffusion into pixels is 3 times as the Floyd's.
3) Stucki error diffusion method: This method reaches to as many pixels as the JJN does but it is quite easy to implement in programming point of view. Because, the pixel values divided by 42 produces a number that can be implemented using bit shifting operators. Similar is the case for Floyd-Steinberg. The divisor '16' can be easily implemented using bit shifting operators.

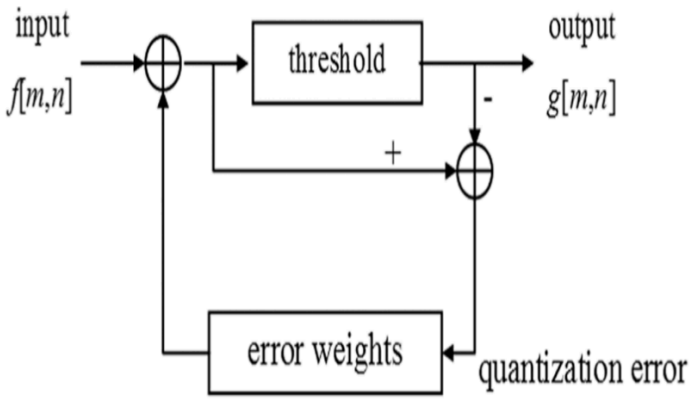


Figure 31:General Block structure for error diffusion

(c) Color Halftoning with error diffusion:
In all previous example we discussed halftoning with respect to greyscale only. When it comes to color images, we apply the same halftoning techniques to each of the three channels. As we know that the halftoning's major application is print media, we generally convert RGB to CMYK and then proceed with each of the channels individually. While to be used as display, the CMYK is converted back to RGB. This scalar property of being able to apply to individual channels without exploiting the relation between the channels is the key trait of digital color halftoning. Here the halftoning is done using error diffusion techniques. One such is the seperable error diffusion method, where the Floyd-Stienberg error diffusion technique is used to quantize each channel. This is a self-intuitive approach to digital color halftoning. In practice we use of much advanced algorithms such as Maximum Brightness Variation criterion (MBVC), and the variation Quadruples algorithm. We'll discuss about them in our later sections of this problem.

APPROACH AND PROCEDURE:

(a) Dithering and Thresholding:

We have to produce binary image for the given input greyscale image using thresholding. The Thresholding techniques are

(i) Fixed Thresholding

(ii) Random Thresholding

(iii) Thresholding using Dithering matrix (I_n).

Boiling down to simple terms, Thresholding is a process to convert each pixel value to black or white based on the threshold value. If the pixel value is greater than the threshold value then that pixel is replaced with the white pixel or if its leeser than the threshold value it is replaced with a black pixel. In the end what we'll have is a binary (0 or 1) image, i.e. Image will consist of only either a pure black pixel or pure white pixel and no intermediate tones.

From this point onwards let T denote the threshold value, F(i,j) represents the input image's pixel and G(i,j) represents the output image's pixel.

Procedure:

- (i) Fixed Thresholding: Fixed Thresholding is the simplest thresholding algorithm. Here we fixate on a threshold value 'T' and we compare each pixel of the input image with the threshold value. If it is greater than the threshold its replaced by a white pixel, else with a black pixel. Mathematically it can be represemted as,

$$G(i,j) = \begin{cases} 0 & \text{if } 0 \le F(i,j) < T \\ 255 & \text{if } T \le F(i,j) < 256 \end{cases}$$

- (ii) Random Thresholding: This is similar to fixed thresholding, but instead of having a fixed threshold throughout the image, we randomly generate the threshold value using the random number generator. For each pixel, a random number is generated within the range [0,256) and this threshold value is used to process the input greyscale image into binary image.

$$G(i,j) = \begin{cases} 255 & \text{if } rand(i,j) \le F(i,j) \\ 0 & \text{if } rand(i,j) > F(i,j) \end{cases}$$

- (iii) Thresholding using Dithering matrix: We see that the above two dithering-halftoning methods are inefficient as they produce adhoc and rigid black and white image. It is always expected for an image to be dithered such that there is a smooth transition in the tones rather than abrupt hard tonal changes. Here the dithering matrix I_2 is given

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

and using the formula given below we construct the I4 and I8 dithering matrices.

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

$I_{4(i,j)}$ =

The 4x4 Dithering matrix:

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

$I_{8(i,j)}$ =

The 8x8 Dithering matrix:

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

The threshold value using which we halftone the image is calculated using,

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

And as usual, every pixel of the greyscale image is read and compared with the Threshold value and assigned with white or black pixel value respectively.

- Procedure for display ready 4 greyscale level image is described in the discussion session of this question. (Please make a note).

(b) Error diffusion method for Halftoning:

(1) Floyd-Steinberg error diffusion method:

Here we apply the error diffusion technique using this 3x3 widow matrix

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

For every pixel we read, we consider its neighbouring pixels that fall within the 3x3 widow area and add the error to the current pixel and redistribute the error to its neighbours. Here for every (i,j) th pixel we read, we find the pixel's proximity to 0 orr 255 and accordingly calculate the

distance between them. This distance is taken the error value 'E' and as shown in the figure below, we add the modified error value to its neighbours as a linear sum.

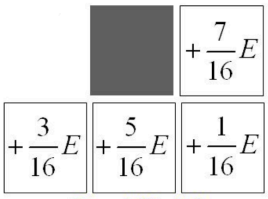


Figure 32: Floyd-Steinberg Error matrix

Like wise we proceed to all (i,j) th pixel in the image. In this problem in particular we use serpentine scanning.

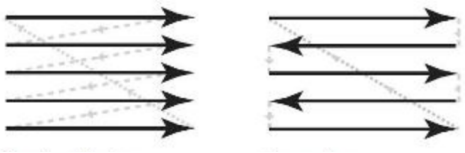


Figure 33: Normal Scanning Serpentine Scanning

That is, once the locator reaches end of row, it starts from the end point of second row and traverses in reverse and continues so on.

(2) Jarvis, Jude and Ninke method of error diffusion:

JJN's error diffusion matrix is shown. In contrast to Floyd-Steinberg's, this method includes more neighbouring pixels in its process.

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

Procedure to implement this method is same as Floyd-Steinberg's. Here its 5x5 matrix instead of 3x3 and scanning is normal rather than serpentine. For every pixel compared, the error is calculated and as mentioned in the matrix, its is added cumulatively to its neighbours.

(3)Stucki error diffusion matrix:

Procedure is exactly the same as the one followed for JJN except for the divisor. Here its 1/42 and there it was 1/48.

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

In methods 2 and 3, once the locator reaches the end of a row, it'll move to the beginning of the next row.

(c) Color Halftoning using Error diffusion:

(1) Seperable Error Diffusion:

The first and the foremost thing to do is to convert the RGB color image to CMYK colorant image. This is done using the formula,

C=255-R ; M=255-G ; Y=255-B

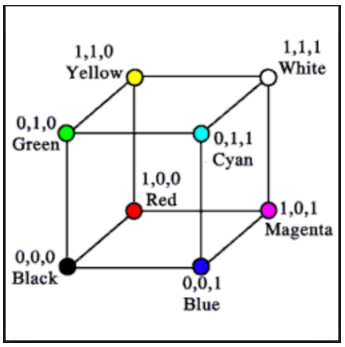


Figure 34: The RGBCMYK cube in RGB scale.

Like in normal error diffusion techniquse where we find whether the current pixel is closer to 0 or 255 (Only one channel), here too we find to which coordinate of the cube is the current pixel close to. The distance can be calculated using any distance norms. We generally use L1 or L2 norm. Here we have used L2 norm.

- The current pixel is read and found to which coordinate of the cube it is closer to.
- Then accordingly to its nearest neighbour, its channel value is made 255 and rest of the channel's value is made 0.
- And the corresponding error value is calculated and it is diffused to its neighnbour using floyd-steinberg's error diffusion matrix.

These three steps are followed for each pixel and the error diffusion process takes place simultaneously for three color channels, but singularly for the each without affecting the other.

(2) MBVQ based error diffusion :

Maximum Brightness Variation Quadruples(MBVQ) overcomes the shortcomings of Seperable error diffusion. Here the RGB-CMYK cube is partitioned into 6 quadrants as shown below.

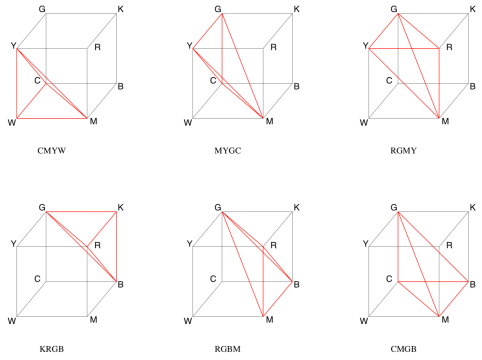


Figure 35:RGM cube's tetrahedral partitions

Here the number of steps involved in halftoning the color image is greater than that in the previous method, but is more optimised.

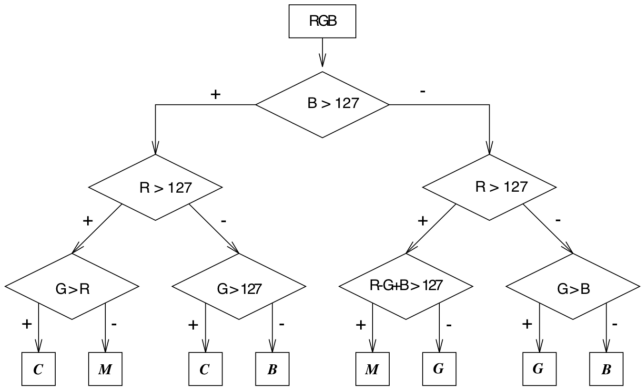


Figure 36:Decision Tree

Given above is the decision tree using which we'll find out, to which tetrahedral partition of the cube our current pixels belongs to. Unlike previous method where we had to find the closest vertex considering all 8 vertices, here we'll hot down to only 4, making our operation computationally faster. The six tetrahedral parts are, (i)CMYK, (ii)MYGC, (iii)RGMY, (iv)KRGB, (v)RGBM, (vi)CMGB as shown in the above figure.

- So the current pixel is read and its corresponding tetrahedral group is found using the if conditional structure.
- Once the group to which the pixel is found, its distance from each of the vertex (here only 4 than 8) is calculated and the nearest neighbour's value are is assigned to all of its channels respectively.
- Then the error value obtained after assignment is diffused to its neighbours through using one of the error diffusion matrices. (Floyd-Stienberg, JJN or Stucki).

Then this process is followed to all other pixels in the image.

##Add in discusion

Then general underlying process of error diffusion is the same, but in previous case we had compared distances for all 8 pixels, but here through region partitioning we compare for only 4 pixels.

EXPERIMENTAL RESULT:

(a)Dithering

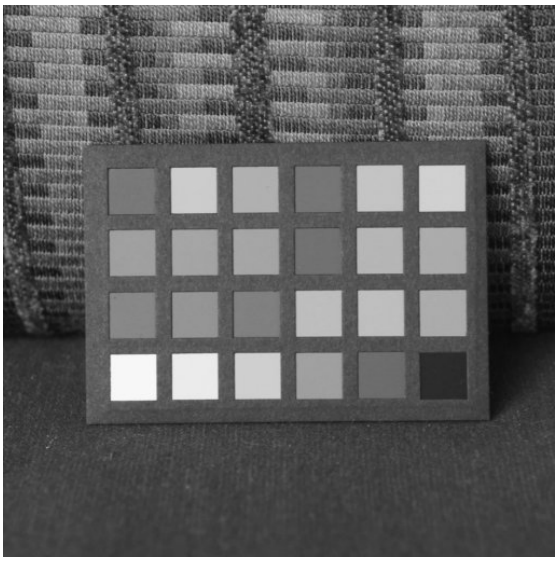


Figure 37: colorchecker.raw (Original Image)

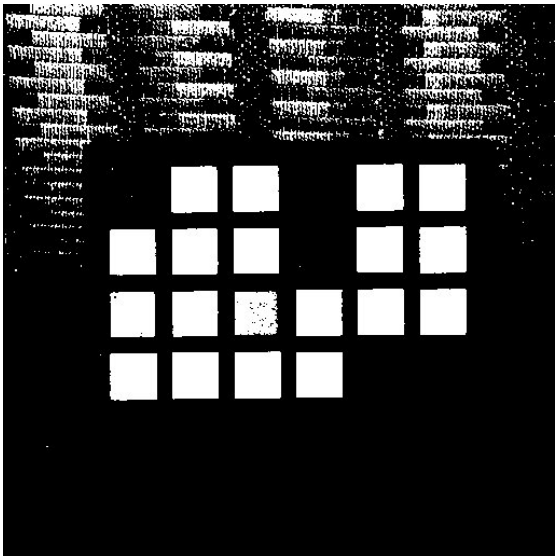


Figure 38: Fixed Thresholding (T=127)

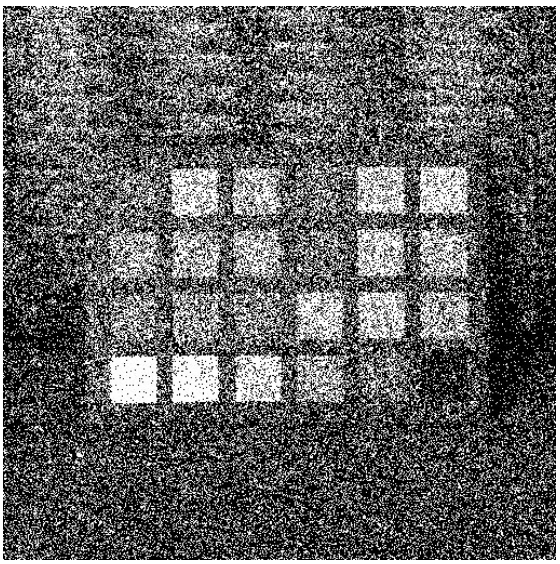


Figure 39: Random thresholding

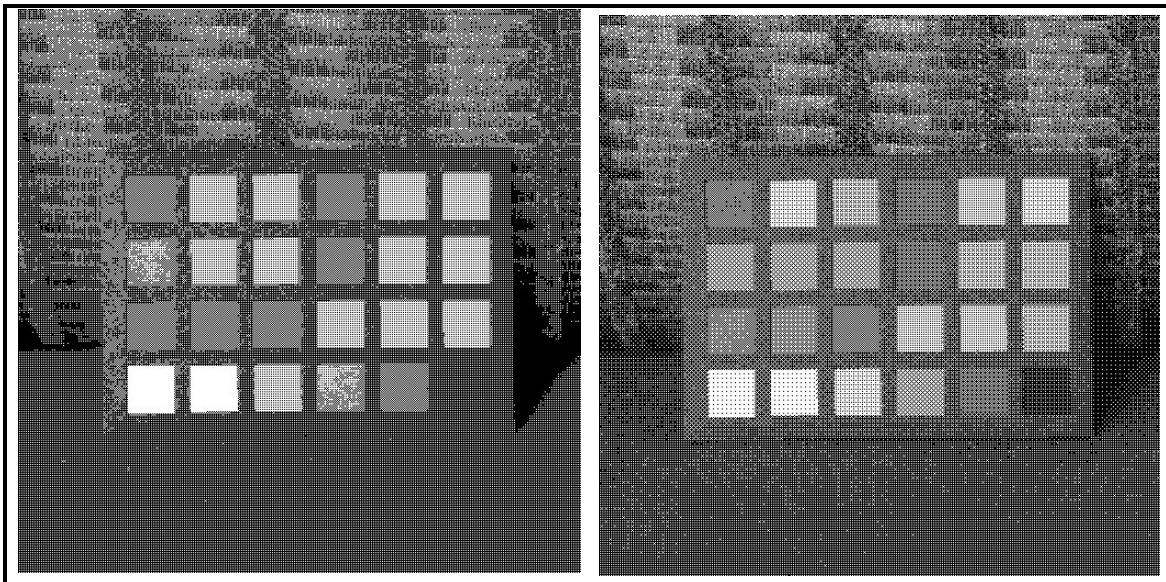


Figure 40: Thresholding using dithering matrix I2(left), I4(right).

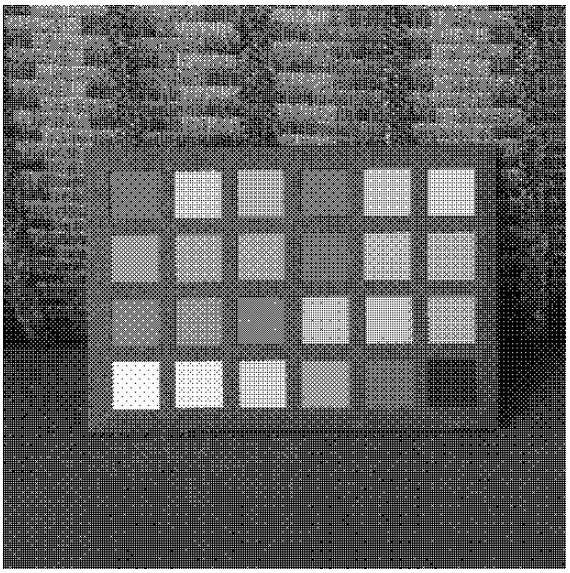


Figure 41: dithering matrix I8

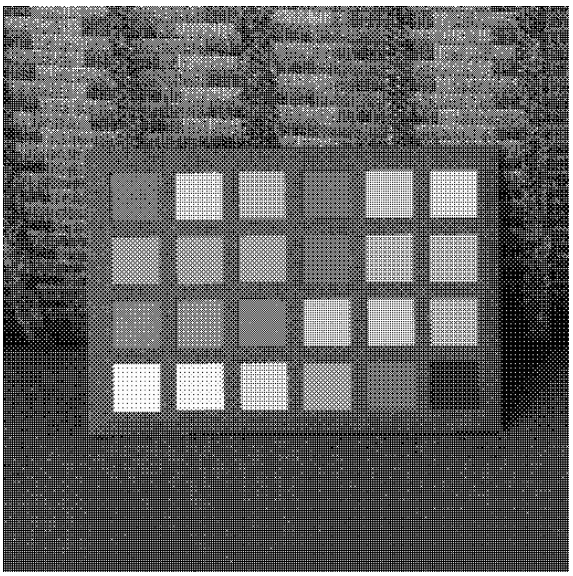


Figure 42: 4 greyscale level, display ready image

(b) Error Diffusion:

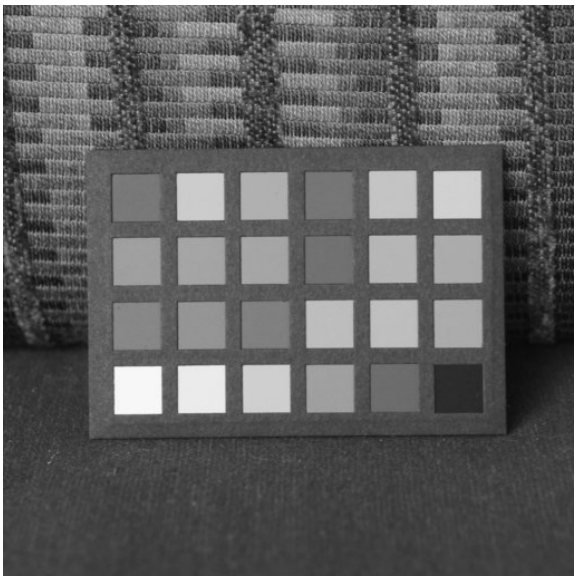


Figure 43: color checker (Original Image)

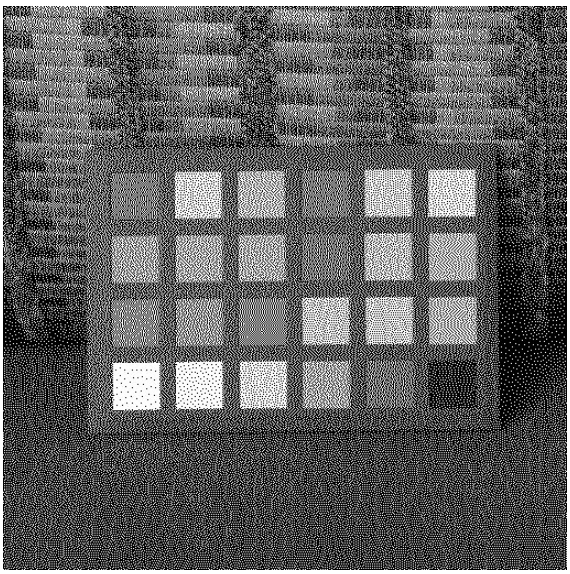


Figure 44: Floyd-Steinberg Error diffussion

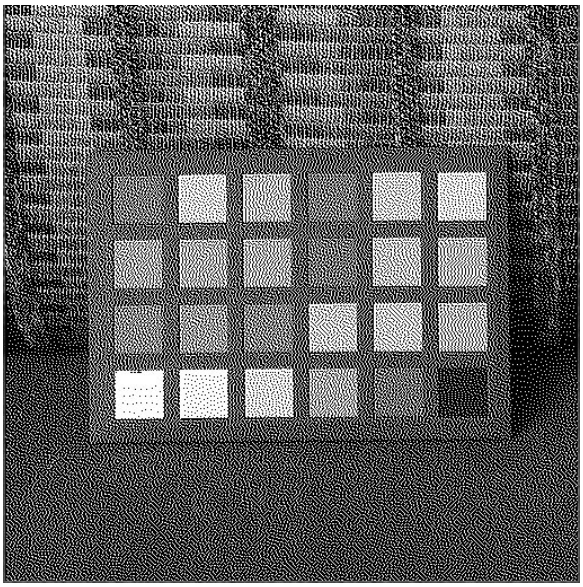


Figure 45: JNN Error diffusion

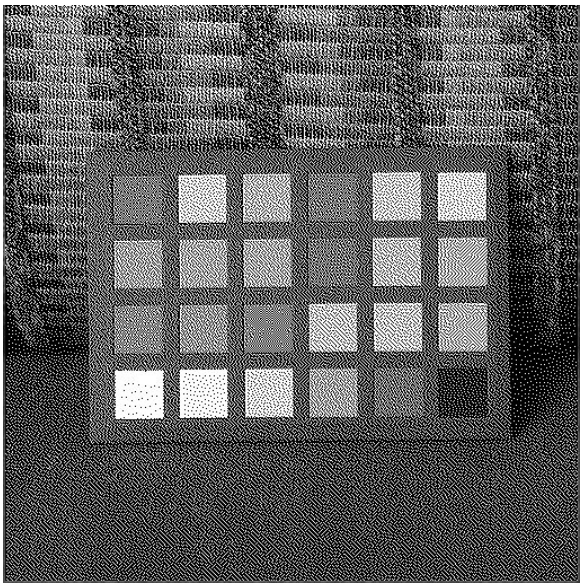


Figure 46: Stucki Error diffusion.

(c) Color halftoning using error diffusion.

Figure 47: Flower (original)

Figure 48: Seperable Error diffusion method

Figure 49: MBVQ error diffusion technique.

DISCUSSION:

(a)Dithering:

This entire concept of Thresholding in dithering works on the same and simple formula, but just that its applied in different variants. We check if the present pixel's color is more close to 255 (white) or 0 (black). And the closest value is assigned to the present pixel. From our different thresholding techniques, we can infer that,

In Fixed thresholding, assume we have patches of pixels with similar color/grey level. Then all the pixels belonging to the patch is converted to the same color and similar with number of other

patches in the image. This results in a patchy, non-desirable output image and especially the originality of the image is lost.
In Random thresholding, we see that we take random threshold value for each pixel. This on the whole makes the picture binary as expexted but the output of the image seems to be too noisy or distorted in tone or the originality of the image is ruptured. In Fixed thresholding at least we can pressume an output and accordingly we can tweek to get a different desired output image. But incase of Random thresholding, there is no definite pattern to follow and apply modification inorder to obtain the output to a desirable extent.

1)In both the above cases, too much of the image's information is lost during thresholding. So inorder to improve the object's retentivity we make use of dithering matrix. This dithering matrix makes sure that the thresholding for dithering is done aptly. In sense, in many of the cases where we quantize the image, we get a layered or patchy tonal image. Dithering matrix method helps to over come this unhealthy effect. Here, when the threshold is applied in accordance to the dithering matrix, the tones are mixed and matched in such a way that we get a smoother finish on the final object than any of the other above mentioned techniques. Here we have used I2, I4 and I8 dithering matrices. It is vivid and evident that as the 'n' increases in In, the final image is more smooth in finish. The halftoned image using I2, I4 and I8 dithering matrices are shown above in the observation section.
2) As the case is provided in question as to quantize the given image to 4 greyscale level considering to create a display ready image, we can quite intuitively use the binary coding technique.
Method (1): We can hard threshold the image. We can split the range as [0,85], [85,170], [170,255] and compute the midpoints for each range. This point will act as the thresholds. Therefore, eventually I will have 4 thresholds.
E.g: The threshold for [0,85] would be 42;
Thus if the pixel intensity is less than this threshold value then quantize it to zero, otherwise if the intensity is above the threshold and below 85 then quantize it to 85. Similarly for the entire range.

Method(2):Since we have to quantize into 4 levels we can make use of two different binary images of the same original image. Two? Because 2^2 gives rise to 4 levels namely,
(0,0) for level 0,
(0,1) for level 1,
(1,0) for level 2 and
(1,1) for level 3. [here, the four levels are (0,85,170,255)].
That is (binary pixel of image 1, binary pixel of image 2) = level.
We can make use of any two binary images, meaning the images produced using different dithering-thresholding techniques. Using a combination of different binary images produced, I find that using I4 and I8 have produced the best output image. I4 and I8 in specific because, from all the above dithered image, I4 and I8 dithered images have been reproduced in much better way.

We see that, this 4 level greyscale image appears much better than any of the above reproduced images. (i) because it has more tones to be used to reproduce the image.

(b) Error Diffusion:

We have seen the dithering techniques and have found out that they are not efficient as the error diffusion processes. The dithered image is halftoned but the level of retentivity of object's informations is very low. Especially in fixed and random thresholding, we can see lots of information loss. In error diffuion techniques, Floyd-Steinberg is one of the most famous error diffusion tyechniques and its is very handy to implement it. Its handyness comes with a price. Though they are better than any of the dithering techniques, when compared to the techniques in error diffusion, they posses a grainy reproduction of the image. The JJN (Jarvis, Jude and Ninke) error diffusion matrix is not quite popiular but it is very powerful in reproducing the halftoned images. It can be seen that, they are less grainy than the floys's. Plus here we take into account more number of neighbours than the floyd=steinberg's. Owing to this, this algorithm was considered complex. Similar windowing to the JJN is the Stucki error diffusion matrix. The image reproduction using stucki looks similar to that of JJN.

Note: Floyd-Steinberg matrix has a divisor 16, so it can be operated using bit shift. Similarly the error values after division by 42 in stucki's are.

Hence Stucki's algorithm for error diffusion is preffered over JJN's.

(c)Color Halftoning using Error diffusion:

(i) Seperable error diffusion: we see that we have obtained a perfectly halftoned color image. It seems to be smoothened because of this halftoning process. This is a very handy technique to apply for color halftoning. Under cover we use/implement floyd-steinberg's error diffusion principle. So it comes with the limitations of floyd-steinberg's method. Here while thresholding it based on its nearest neighbours, we see that it computes for all 8 vertices of the RGBCMYK cube. Though this method is robust, it computationally takes a very long tine to process.

(ii) MBVQ error diffusion technique: From the output obtained, figuratively I could find not much of a difference between them. But algorithmically, MBVQ is way better and optimized algorithm to be used and it is computational friendly too. Here to compute the nearest neighbour, we compare with 4 vertex only, that is an advantage in terms of computing time. This is because of the fact that we segregate the pixels into 6 tetrahedrals regions and then only we proceed with the normal halftoning function.

PROBLEM 3: MOPRPHOLOGICAL PROCESSING

AIM:

To implement the given 3 morphological processing operations, namely shrinking,thinning and skeletonizing.

(a): Shrinking:

(i) To count the total number of stars in the image.

(ii) To figure out how many different star sizes are present in the image and its corresponding frequency.

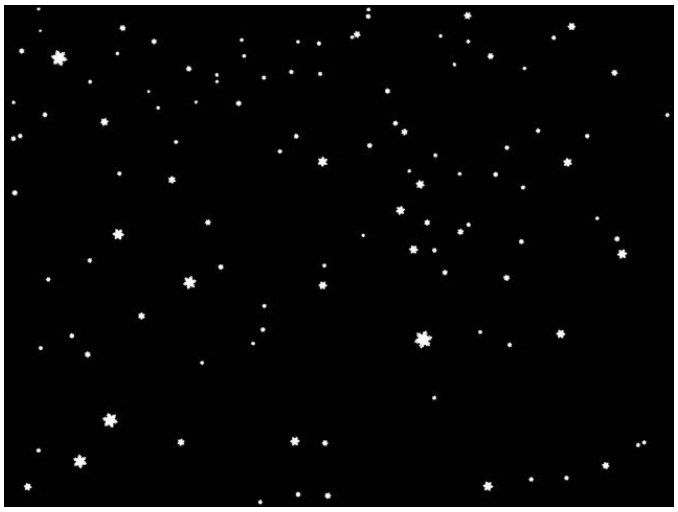


Figure 50: stars.raw

(b): Thinning:

To apply the thinning filter to the given image, Jigsaw_1.

Figure 51: Jigsaw_1

(c): Skeletonizing:

To apply the skeletonizing filter to the given image, Jigsaw_2.

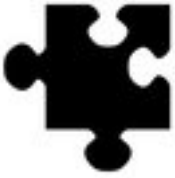


Figure 52: Jigsaw_2

(d): Counting game:

(i) To count the number of pieces in the image board. (Without the pieces being unique)

(ii)To count the number of unique pieces in the board image. Here consider the uniqueness of the piece.

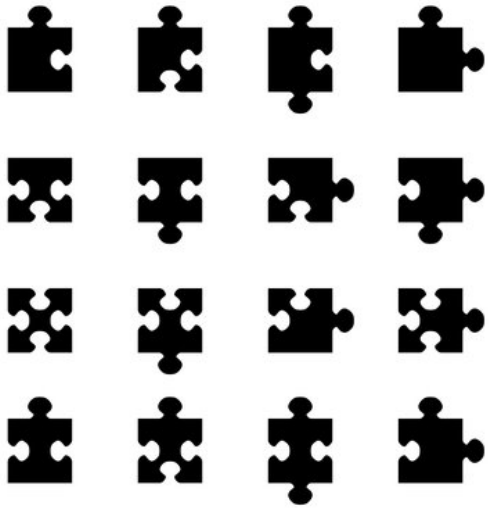
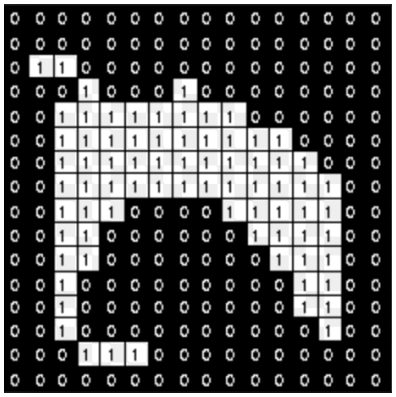


Figure 53:Board

ABSTRACT AND MOTIVATION:

Morphological Processing in Digital images is a collection of operations that are related to the morphology of the image. i.e. form or shape or other such attributes of the image. The use of morphological image processing is mainly due to the fact that binary images or converting normal continuous images to binary results in numerous imperfections in the reproduced image in the form of distrotion due to noise or texture. The aim of morphological image processing is to remove these imperfections without affecting the shape or form or structure of the given image.

Example of a binary image(Normalized),



or any image obtained in Q 2.

Figure 54: Binary Image

Morphological processing deals with relative ordering of the pixels in the image rather than their numerical values. In short, it deals with “where is it ” than “what or How much is it”. This process

uses a window (most often it us a 3x3 window in our use, but any odd N x N can be used). Using this window and the pattern table, we create another binary reference image denoting the 'hit' or the success condition of the pattern. The pattern table is a list of patterns or reference procedures that are used in the creation of the hit & miss image. Moreover they are a guide to check if a particular form/pattern is followed. The window of the present pixel (the pixel along with its neighbours) is compared with each of the patterns from the table and marked a hit if it matches any. Likewise we have different pattern tables for different morphological functions. Three of the basic morphological processes will be implemented here. Namely, Shrinking, Thinning and Skeletonizing.

Shrinking: Shrinking and Zooming are 2 important aspects of digital image processing.Shrinking intuitively means to reduce in size. Here too, it is a process where an object in the background is shrunk to its lowest level of occupancy state for existance. i.e lowest possible pixel area without dissapearing.

Thinning: Thinning is a process of shrinking whilst preserving the topological aspects of the image, while shrinking only preserves the existance of the image. Here the object shrinks along its outer boundary without breaking down into fragments. Thinning bring the object to 1-pixel thickness.

Skeletonizing: Skeletonizing is an extended or modified version of thinning. Where in thining only pixel width reduction is the main goal, skeletonizing is the process of removing pixels converting the binary image into a skeletal structure preservig its form and extent.

In the counting game, we'll come across a topic called "Connected component Analysis". In an 2-D image, connected components are the clusters of pixels that have same binary values that are considered connected either on 4-connectivity or 8-connectivity rule. This component analysis acts as a surrogate funtion in finding the number of connected objects in an image by labelling them.

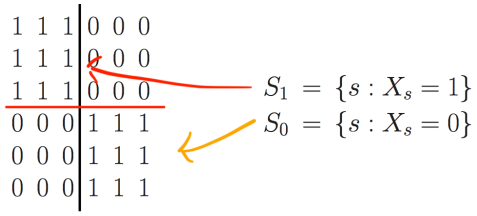


Figure 55: Connectedness

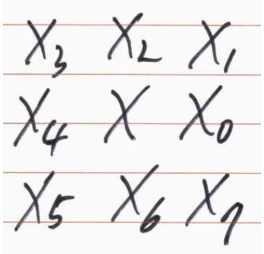
In 4 connectivity rule: s1 and s2 are not connected and in 8 connectivity rule they are connected. For our ease, here we'll be considering 8-connectivity rule. **…….....(Assumption)**

APPROACH AND PROCEDURE:

(i) Shrinking:

- First we'll have to make a note that these morphological operations are possible only in binary images. So we will convert the given image to binary (0's and 1's) image. A Hit&miss image variable whose size equal to the image size is created.

- Using loops and conditional blocks, we detect the white pixels only.(Here the object is white and the background is black. So, we neghlect the black pixels while reading). Every time a white pixel is read, a window of its neighbours is created in the form as shown below.



This array of 8 units ranging from X_0 to X_7 (X denotes the current pixela and is not included in the pattern as it is 1 by default.)is compared with the given pattern table. First we create the Hit&Miss array using the conditional patterns.

- These conditional patterns can be compared as such by element-wise comparison or the entire array's information can be converted into a binary code (Because the image is itslef binary) and compared. Here using proper conditonal structuring for bonds value we compare the pattern and if matched we '1' is returned else '0' is returned. Likewise this procedure is followed for all the object's pixels and the Hit matrix is generated.
- This is just step 1 towards our shrinking process. Once we generate the Hit matrix, we use this matrix and follow the same windowing technique as above to compare our pattern in the hit matrix to those Unconditinal patterns in the pattern table or the lookup table. Here, if the hit matrix pattern and the unconditional pattern matches, we over-write our actual binary image with a '0' and if not, we leave the actual image's pixel as it is. At the end of this stage we'd have shrunk our image possibly by 1-pixel's measure.
- These two steps are performed itertatively until we reach point where we can't shirnk the image, possibly, further. In general, even if we repeat this process for infinite number of times, our final image will still remain the same. For instance, for an image that is shrunk to an unit pixel's size, and if still we continue to follow the above two steps, in the step 1 where we generate the Hit&Miss matrix, we'll never get a hit mark. Hence that pixel never goes into check for the unconditional patterns and hence there will be no data over-writing on it.

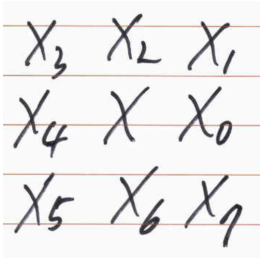
For measuring the star size:

- Here, after every loop, i.e after the pixel has gone through both the conditional and unconditional mask patterns, we use a separate function to count the number of stars that are of unit size. By tracking the number of unit sized-object in each iteration, we can find out the number of stars for each size. The data retrieved here can be used to plot the histogram for the star-sizes.

(ii) Shrinking:

- Like shrinking, thinning also follows the same/similar steps from the perspective of coding.
- Step 1: For every non-background pixel, we window out the pattern followed by it and its neighbours and compared with each of the patterns in the pattern table. Accordingly the Hit&Miss matrix is generated.
- Step 2: From the generated hit matrix we make comparisons with the unconditonal patterns and likewise we over-write new information or pass.

We came across a term called 'Bond', it says how connected the current pixel is to the object, the bond value varies from 1 to 12. The bond value for its neighbours is shown below,



1	2	1
2	X	2
1	2	1

Say X has neighbours X0 and X1, bond value of X = 1(1)+1(2) = 3.

The pattern table is self instructive. For Shrinking and thinning we use same unconditinal patterns and the conditonal patterns are bond marked so that we make use of appropriate pattern masks only. Example, the thinning pattern masks are from bond value 4 through 10 only and for shrinking its 1 through 10 only.

(iii) Skeletonizing:

- The steps to be processed are the same but, we make sure that we follow and check the right patterns only. In skeletonizing, the bond value to be taken into account is 4 through 11 excluding 5.
- The unconditional pattern masks to be followed are quite different from those that were used for shrinking and thinning.

(iv) Counting game:

(a) Count the number of objects present in the picture:

Here shrinking and thinning would have been an imperative option to follow. But we follow the connected component analysis as it is favorable over the other.

- Check if the pixel we read is an object pixel or background pixel. Proceed further for object pixels.
- Check is the pixel is already labelled or it is yet to be labled.

Consider the windowing technique as explained in the previous subproblems.

- (i) Un-labelled new object pixel: Use a counter variable to label the new pixel.Everytime we encounter a new unlabelled pixel, initilize the counter value as the pixel's label and

increment the counter. And check all of its neighbours and to those that belong to an object, lable them with this lable mark.

- (ii) More efficient way is check all of its neighbours if they are labelled. If so find the min value of the label in the current pixel's vicinity and change all the object pixels to the new minimum value lable.

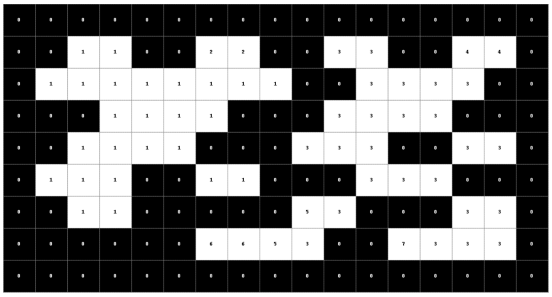


Figure 56: Step (i)

- (iii)After all new pixels are labelled once (it is not the final label value of the object as all the pixels belonging to the object don't have the same label value.) when you follow step two, or by continually iterating throughout the pixel, we take into account all of the neighbouring pixel's value and find the label value that is minimum and assign that value to all the object neighbours present. By this way we can converge to a possible min value label for each object. At the end of iterations, we can see that all the pixels belonging to a particular object have the same label value.

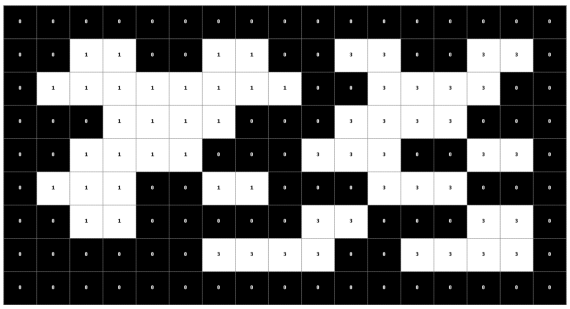


Figure 57: Step(iii)

(b) Counting the number of uinque objects in the given image:

The reason why I chose connected component analysis over shrinking is this part of the question. Here we are asked to find the number of unique objects present in the given image. Since we have all the objects already labelled, it is more convinient to proceed with this question.

Before proceeding with the procedure, the following assumptions are made**,……..…(Assumption)**

- All the objects in the image are considered to be strictly scaled, i.e. the objects with different dimensions are considered to be unique.
- The rotated and the reflected versions of the objects are not considered to be unique.

I have used a combination of two simple strategies in action to find out the number of unique objects.

I: Assuming scaling is considered for uniqueness. Anyhow the objects here are of similar scaling only.

- First we figure out the following attributes of the object in the image,

(i) Number of pixels the object holds, (ii) the span of the object .i.e the x-range and the y-range of the object and (iii) its min and max dimensions of the particular object. (This (iii) comes in handy to selectively the compare the individual objects).

- Then I compare the pixel count, x-range and y-range of the each object with every other object. (Here the ranges are cross compared, so that object that are rotated are accounted for similarity). All those objects that have its attributes (x-range, y-range and count) completely in contrast are consider unique. [Objects that don't have similar dimensions or similar pixel count in them are obviously not similar in structure]. For tracking this, we introduce an error counter and a tolerance value for error. [Tolerance, because, while converting the image to binary, there might be certain imperfections in thresholding the object].
- From this, we can sort out the completely unique objects and those that we are uncertain about.

II: This is the main procedure used to figure out the similarities between the normal and rotated/reflected forms of the same object.

- Both in rotation and reflection, we can compare a normal object with its rotated/reflected or both, just by changing the array indexing. As discussed earlier, we'll use an error counter and a tolerance variable to check if the objects are similar. Error is checked by comparing each individual pixel of both the objects. If they match (either both the pixels are labelled or not labelled), we proceed further and if they don't (both the pixels have different value) we increment the error counter. Once the entire iteration is done, and if the error value obtained is below the tolerance level, we say that the object is similar and for dissimilar objects, the error rates will be very high.

Note: each object is compared with every other object's normal form, rotated,reflected or any of the combinaiton of these. And still we find a match between them, we say that they are similar and if they arent they are labelled as unique object.

EXPERIMENTAL RESULTS:

(i)shrinking:

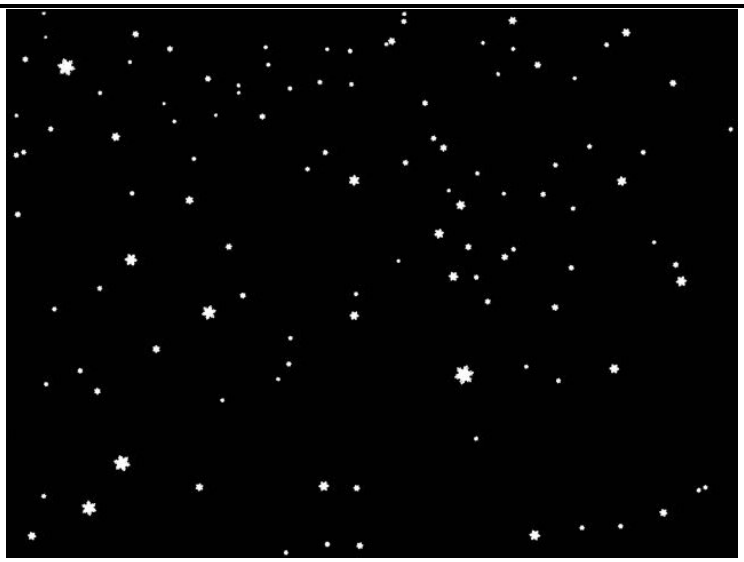


Figure 58: Stars (original)

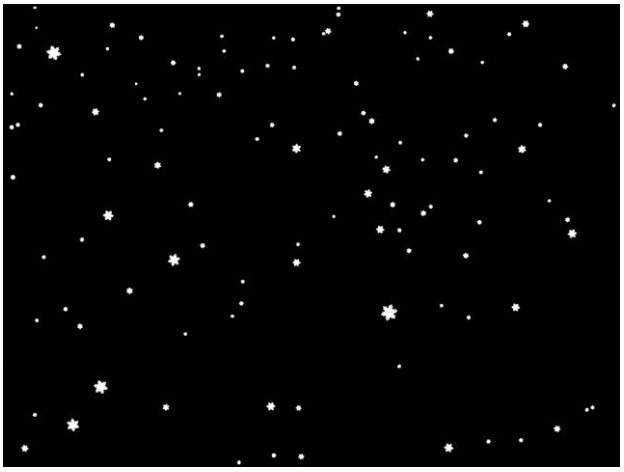


Figure 59: Stars Binary

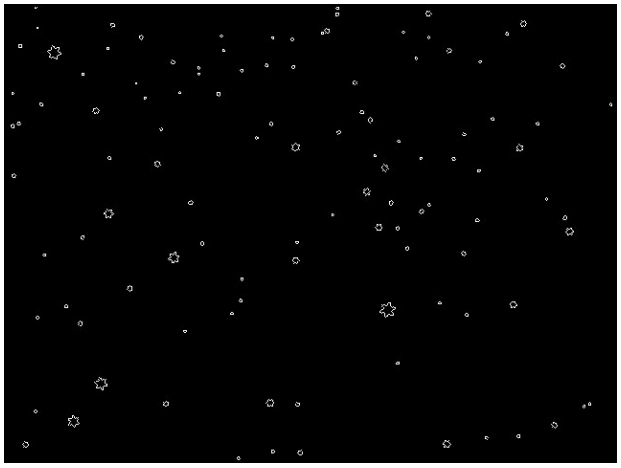


Figure 60: Hit mask after iteration 1.

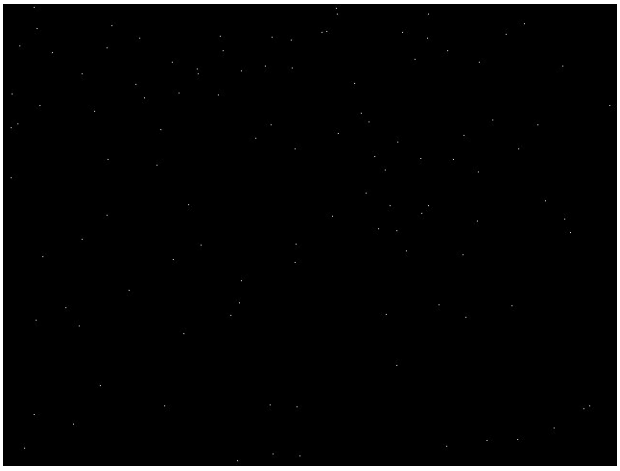


Figure 61: Stars shrunk.

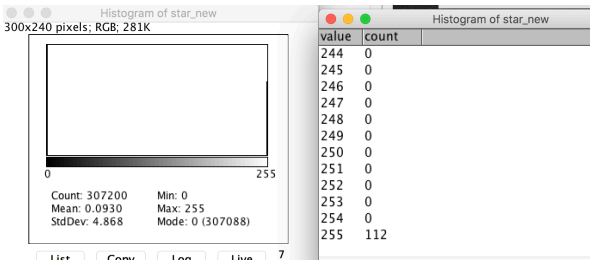


Figure 62: Showing number of stars

```
guest-wireless-207-151-060-088:test rickerish_nah$ ./s /Users/rickerish_nah/Docu
ments/test/stars.raw /Users/rickerish_nah/Documents/test/star_new.raw
Showing number of stars for each size in the order of increasing size.....
at i=0 #stars is:7
at i=1 #stars is:12
at i=2 #stars is:38
at i=3 #stars is:14
at i=4 #stars is:22
at i=5 #stars is:9
at i=6 #stars is:2
at i=7 #stars is:3
at i=8 #stars is:2
at i=9 #stars is:1
at i=10 #stars is:0
at i=11 #stars is:1
at i=12 #stars is:0
at i=13 #stars is:0
at i=14 #stars is:1
at i=15 #stars is:0
at i=16 #stars is:0
at i=17 #stars is:0
at i=18 #stars is:0
at i=19 #stars is:0
```

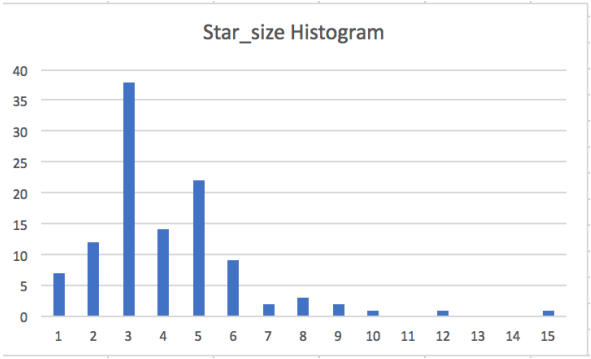


Figure 63: Showing stars of different sizes and corresponding

(ii) Thinning:

Figure 64: Jigsaw_1 (original)

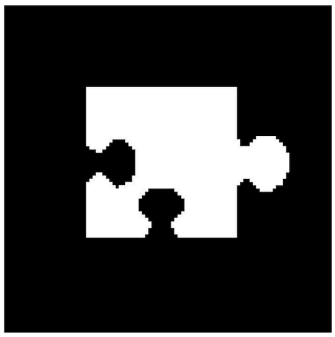


Figure 65: Jigsaw_1 Binary (Inverted)

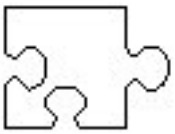


Figure 66: Hit mask after first iteration. (actual)

Figure 65: Jigsaw_1_thin (inverted)

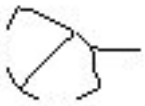


Figure 66: Jigsaw_1_thin (actual)

(iii) Skeletonizing

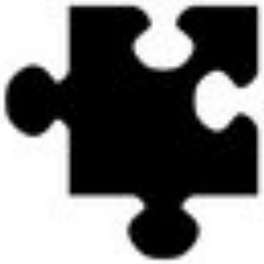


Figure 67: Jigsaw_2 (original)

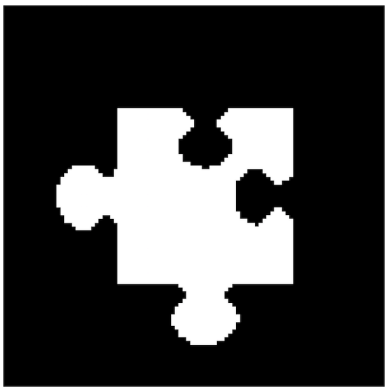


Figure 68: Jigsaw_2 binary (inverted)

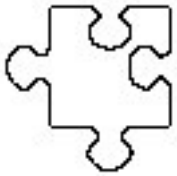


Figure 69: First iteration hit mask.

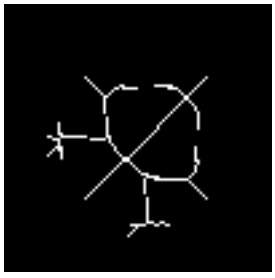


Figure 70: Jigsaw_2_skeleton (inverted)

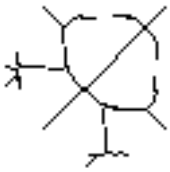


Figure 71: Jigsaw_2_skeleton (actual)

(d) Counting Game:

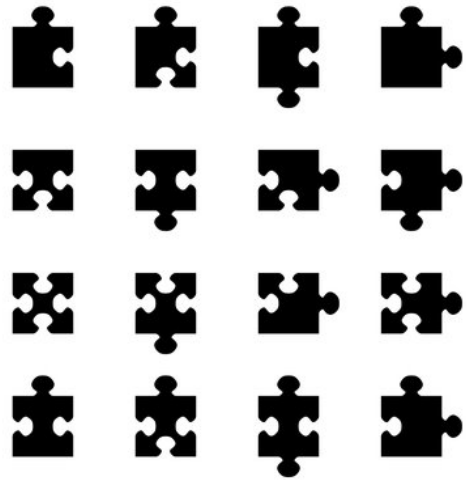


Figure 72: Board (original)

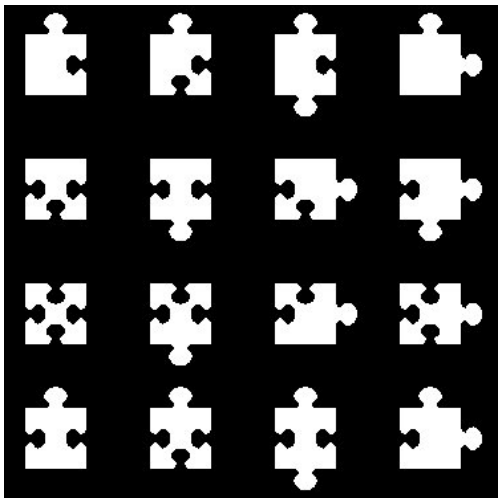


Figure 73: Board binary (inverted)

```
guest-wireless-207-151-060-088:test rickerish_nah$ ./p /Users/rickerish_nah/Documents/test/board.raw /Users/rickerish_nah/Documents/test/puzz.raw
Actual Lable:
        :14     :15     :17     :13     :33     :34     :35     :32     :62     :63     :60     :64     :88     :89     :90     :87
Modified Lable:
        :1      :2      :3      :4      :5      :6      :7      :8      :9      :10     :11     :12     :13     :14     :15     :16
```

Figure 74: Labelled objects. Number of objects present in the board.

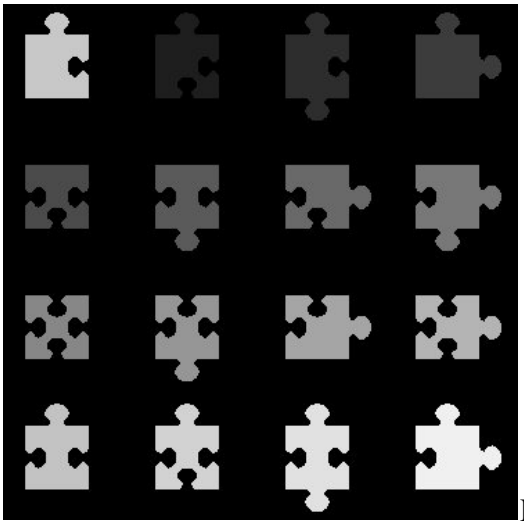


Figure 75: Showing individual objects in different shades.

```
at I:1   count=2166     X_range:61      Yrange:45       Xmax:67 Xmin:6  Ymax:61 Ymin:16
at I:2   count=2013     X_range:61      Yrange:45       Xmax:67 Xmin:6  Ymax:155        Ymin:110
at I:3   count=2385     X_range:77      Yrange:45       Xmax:83 Xmin:6  Ymax:249        Ymin:204
at I:4   count=2539     X_range:61      Yrange:61       Xmax:67 Xmin:6  Ymax:359        Ymin:298
at I:5   count=1644     X_range:45      Yrange:45       Xmax:161        Xmin:116        Ymax:61 Ymin:16
at I:6   count=2019     X_range:61      Yrange:45       Xmax:177        Xmin:116        Ymax:155        Ymin:110
at I:7   count=2018     X_range:45      Yrange:61       Xmax:161        Xmin:116        Ymax:265        Ymin:204
at I:8   count=2388     X_range:61      Yrange:61       Xmax:177        Xmin:116        Ymax:359        Ymin:298
at I:9   count=1482     X_range:45      Yrange:45       Xmax:255        Xmin:210        Ymax:61 Ymin:16
at I:10  count=1858     X_range:61      Yrange:45       Xmax:271        Xmin:210        Ymax:155        Ymin:110
at I:11  count=2010     X_range:45      Yrange:61       Xmax:255        Xmin:210        Ymax:265        Ymin:204
at I:12  count=1853     X_range:45      Yrange:61       Xmax:255        Xmin:210        Ymax:359        Ymin:298
at I:13  count=2008     X_range:61      Yrange:45       Xmax:349        Xmin:288        Ymax:61 Ymin:16
at I:14  count=1854     X_range:61      Yrange:45       Xmax:349        Xmin:288        Ymax:155        Ymin:110
at I:15  count=2227     X_range:77      Yrange:45       Xmax:365        Xmin:288        Ymax:249        Ymin:204
at I:16  count=2380     X_range:61      Yrange:61       Xmax:349        Xmin:288        Ymax:359        Ymin:298
HariKrishnas-MacBook-Pro:test rickerish_nah$ clear
```

Figure 76: Attributes of each labelled object.

```
guest-wireless-207-151-060-088:test rickerish_nah$ ./p /Users/rickerish_nah/Documents/test/board.raw /Users/rickerish_nah/Documents/test/puzz.raw
Actual Lable:
        :14     :15     :17     :13     :33     :34     :35     :32     :62     :63     :60     :64     :88     :89     :90     :87
Modified Lable:
        :1      :2      :3      :4      :5      :6      :7      :8      :9      :10     :11     :12     :13     :14     :15     :16
Number of Objects Indentified is :16


Identifying Objects:......


    Completely Unique Object:1
    Completely Unique Object:3
    Completely Unique Object:4
    Completely Unique Object:5
    Completely Unique Object:9
    Completely Unique Object:15
    Matches found (Upside down):6   13
    Matches found (Upside down):7   11
    Matches found (Rotation):2      7
Number of objects Uniquely Identified is: 10
guest-wireless-207-151-060-088:test rickerish_nah$
```

1 match Unique Object:8

1 match Unique Object:16

2 match Unique Objects:10
2 match Unique Objects:12
2 match Unique Objects:14

Figure 77: Final output showing Uniquely identified objects.

DISCUSSION:

All these morphological processing techniques are of very great help in the revolution of Digital image processing. They are mainly used to remove imperfections in the digital binary images. There are so many techniques such as dilation, erosion, shrinking, etc. We have learnt and implemented 3 main techniques, namely, shrinking, thinning, skeletonizing.

Shrinking: At each stage, an outer contour of the object is seen to be removed, therby shrinking in size. As the result of shrinking, the stars can be seen as a pixel dots on the image. They look like spec of unitary pixel in the image. Once shrinking for all the stars in the image is done, we check for the number of non zero pixels in the image. Thereby we obeserve the fact that there are 112 stars in the object. The star sizes are displayed and plotted as a histogram as shown in figure 63. At 15th iteration all the objects (star) in the image becomes unit pixel size.

Thinning: The final output for thinnig is shown. Unlike the star image where the background is black and object is white, here the background is white and the object is black. So we have to

rephrase all the conditions to black:object from white:object of previous case. Or it is more convinient to convert the image to a black background image and perform thinning operation and at the end to invert it back to the normal condition. Here we can see that during each iteration, all the boundry pixels are removes as an effect of thinning process except for those that are critical to hold the object from breaking.

Skeletonizing: We have obtained the output for skeletonizing in the 17 th iteration.

Counting Game: I had assumed scaled object to be a unique object. Therefore I check the uniqueness of the objects in the image retrospectively. Firstly I've obtained the possible unique objects through comparation sampling. That is to check if the each object's attributes match with another for uniquness (Obtained 6 unique objects – 1,3,4,5,9,16)(2 group matches (8 and 16, 10,12 and 14)). Then updated my label list to be checked of unique objects. Later, considered normal position and reflection of the object and found matches and found match for (6 and 13,11 and 7). And the last object 2 was matched with 7 while checking for rotation.All of this can be implemented using rotation and reflection comparisons through iterative means.

REFERENCES:

[1] http://www.corrmap.com/features/homography_transformation.php
[2] http://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf.
[3] http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/
[4] https://en.wikipedia.org/wiki/Connected-component_labeling
[5] HW attachments.