University of Southern California

Viterbi School of Engineering

Introduction to Digital Image Processing

2018 Spring EE569

Homework #2

Hang Dong

Student ID: 8151308667

donghang@usc.edu

February 22,2018

Problem 1: Geometric Image Modification

(a) Geometrical Warping

1.Abstract and Motivation

Scaling, translation and rotation are the basic geometric transformations of images which are applied in 3D vision technology, it can be further classified into affine transformation and perspective transformation. Affine transformation is usually modeled by homographic character. There are two main operation: vector space conversion and pixel value assignment in the corresponding position between original and output image.

2.Approach and Procedures

Image warping is an application of affine transformation, just do point mapping without changing color. It can be described as the following equation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where (x, y) is the original image pixel position and (u, v) is the corresponding pixel position in new image. T is the transformation matrix, including translation, scaling and rotation, it can be further expressed as:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore we need to pick three pairs pixel to establish equation to solve unknown parameters in transformation matrix. In general, there are two transformation direction: forward mapping and backward mapping. In general, backward mapping is more effective than forward mapping.

In this problem my solution of warping worked as follow steps:

#1: Separate the original image into 4 parts

#2: In each part, apply Pythagorean theorem to calculate the rate of resize of every row.

#3:According to the rate calculated in #2 and Interpolation method to build up new image of warping one pixel by another.

In this problem my solution of reverse warping worked as follow steps:

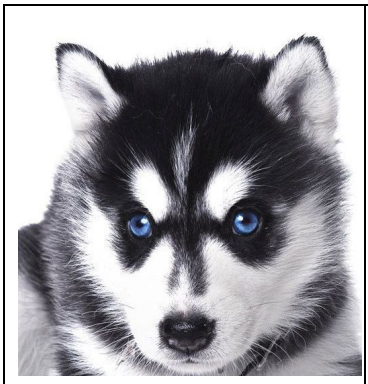
#1: Separate the original image into 4 parts

#2: In each part, apply Pythagorean theorem to calculate the rate of resize of every row.

#3:According to the rate calculated in #2 and Interpolation method to build up new image of reverse warping one pixel by another.

3.Experimental Results

Result of puppy

		
Puppy original	**Puppy warping**	**Puppy reverse**

Sheet 1. Result of puppy

Result of tiger

		
Tiger original	**Tiger warping**	**Tiger reverse**

Sheet 1. Result of tigger

Result of panda

		
Panda original	**Panda warping**	**Panda reverse**

Sheet 1. Result of panda

4.Discussion

By comparing the original images and reverse images of these three images its obvious that they are so similar but the reverse images are little wide than original images.

(b) Homographic Transformation and Image Stitching

1.Abstract and Motivation

In computer vision, the homograph of a plane is defined as a projection mapping from one plane to another, it usually contains more than one views of the same planar surface, like two cameras share the same center but at the different visual angle. We often use homographic transformation to do image stitching to create panorama image. The 2D point (x, y) in the image can be expressed as a form of 3D vector $(x_1,\ x_2,\ x_3)$, in which $\mathrm{x} = x_1/x_3, \mathrm{y} = x_2/x_3$. It is called the homogeneous expression of the point. The homographic transformation is similar to affine transformation which has a 3*3 matrix H. Although it has 9 unknown elements, the scale parameter is arbitrary. This means that there are 8 unknown parameters in it, we need to find four pairs of corresponding pixels to solve them. Typically, the homographic matrix can be estimated by the feature matching between original and output images. The operation can be described as the following equation:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $h_{33} = 1$, (x, y) is the pixel position in original image and (x', y') is that of new image.

2.Approach and Procedures

In this problem, we have three images of the same object from different vision angle. We need to find the mapping relationship from the overlapping part. Therefore, we select four pairs same object pixels that on the boundary of left and middle image, and then do the similar thing between the middle and right image. The object's coordinate in the middle image is that of the output image, which is translated by a constant value from the input image, the coordinate in the left and right image is original ones. From these one-to-one matching relationship, we can build functions and solve them in MATLAB to get two homographic transformation matrices H_1, H_2. Then we can use the matrices to map every pixel in original left and right image to the stitching output image. In some pixel, it will lose some information of the input image, so we need to

do some kind of interpolations like using the surrounding pixels value to estimate the center one. Finally, we get the stitching image of three input.

Algorithm steps as follow:

#1: Using the formula $P_2 = HP_1$ and key points from two different camera viewpoints to build up the H matrix.

#2: Using the H matrix to build up homographic image.

3.Experimental Results

According to the key points I found and the calculation results of Matlab to build up H matrix.

H_1(the H matrix of left.raw and middle.raw)

$$H_1 = \begin{matrix} 2.0567 & 0.0366 & 87.93 \\ 0.8774 & 1.4031 & 371.5986 \\ 0.0011 & 0.000014 & 1 \end{matrix}$$

H_2(the H matrix of right.raw and middle.raw)

$$H_2 = \begin{matrix} 0.3029 & -0.0384 & 709.5805 \\ -0.586 & 0.855 & 527.966 \\ -0.000715 & -0.000042 & 1 \end{matrix}$$

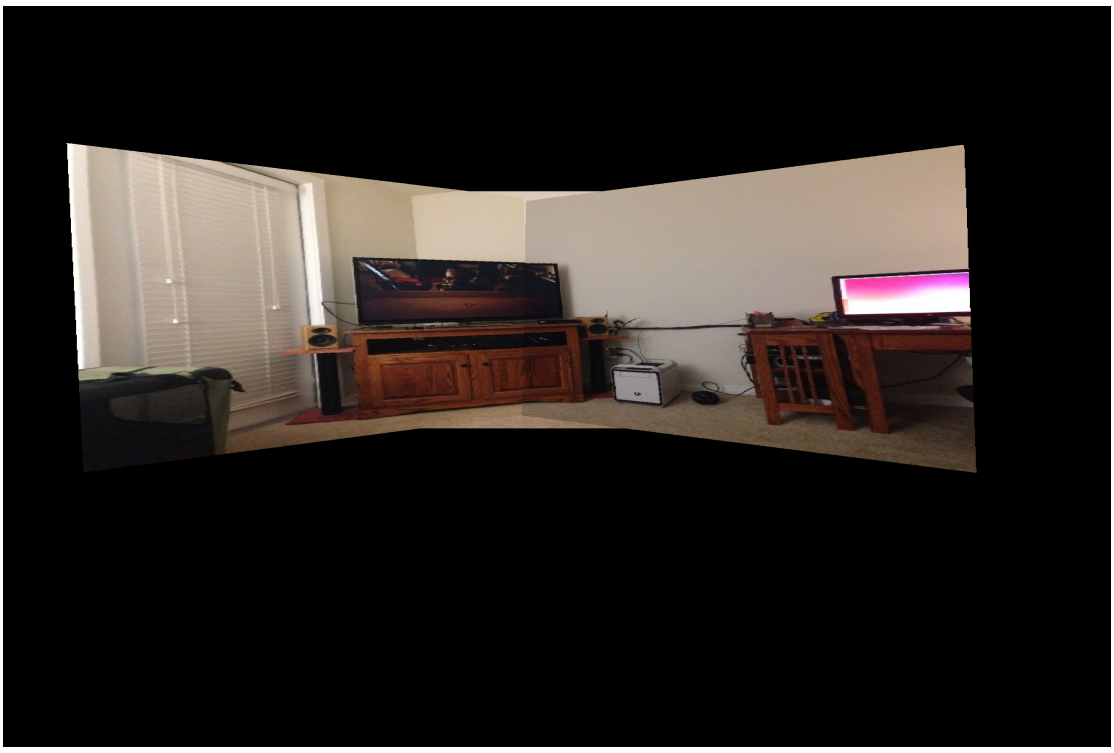


Image1. Homographic Transformation and Image Stitching result

4.Discussion

1) How many control points were used? What if we use more than four control points?

To build up the H matrix, I need at least four points for two images. So in this problem I need 8 points to combine these three images.

2) How did you select control points? Clearly specify your observations and strategies.

To get an exact homographic image, I choose the key points distributed like rectangular and not in same rows and columns. If choose the points that nearby each other, the results will be very bad and even can not accurate.

For example, the key points between left and middle images are reds. The key points between right and middle images are greens. As the image show all points distributed as a rectangular.

Image 2. Key points in middle image

Problem 2: Digital Half-toning

Digital half-toning technology is based on human visual characteristics and image coloring characteristics, and use mathematics, computers and other tools to achieve optimal reproduction of images on binary color devices. Usually, the basic pixels in

half-toning image only have black or white two status, it use the exposure point arrangement and size to realize image replication, it reflects the brightness level and color contrast of an image. In printing, the colors and dilation of the objects are expressed on these tiny exposure pixels, which can produce many gray-scale illusion to human eyes. There are three basic way to get half-toning image: dithering, error diffusion and iteration.

(a) Dithering

1.Abstract and Motivation

It is a simple way to realize half-toning. Generally, it generates a threshold for each pixel and compare the value, if current pixel value is bigger, put 255 in the half-toning image, otherwise, output 0. There are three ways to set threshold, fixed number, random number and dithering matrix.

2.Approach and Procedures

(1) Fixed thresholding

The threshold is fixed and same for each pixel in this method. Since there are 256 gray-scale levels, we always choose $\mathrm{T} = 127$ as the threshold. Through the loop, we check every pixel value and assign 0 or 255 to the new output image. It can be described as the following equation:

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

(2) Random thresholding

It is the similar procedure to the fixed thresholding method, the only difference is that we generate different random threshold value $rand(i,j)$ for each pixel. It can be described as the following equation:

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

(3) Dithering Matrix

Instead treating every pixel totally randomly, we generate a regular dithering matrix pattern to check original image pixels recursively. The number in the matrix reflects the occurrence probability of the same position in the output image. The most frequently used matrix is Bayer ordered dithering matrix. It is a 2*2 matrix and it can generate different size dithering matrices by the following equation:

$$I_2 = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

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

After getting the dithering matrix, we can obtain the threshold matrix T by the following equation:

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

and N is the matrix size. Before applying the dithering matrix, we need to get the normalize input image, and then do the same work as above, it can be expressed as the following equation:

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

3.Experimental Results

(1) Fixed thresholding

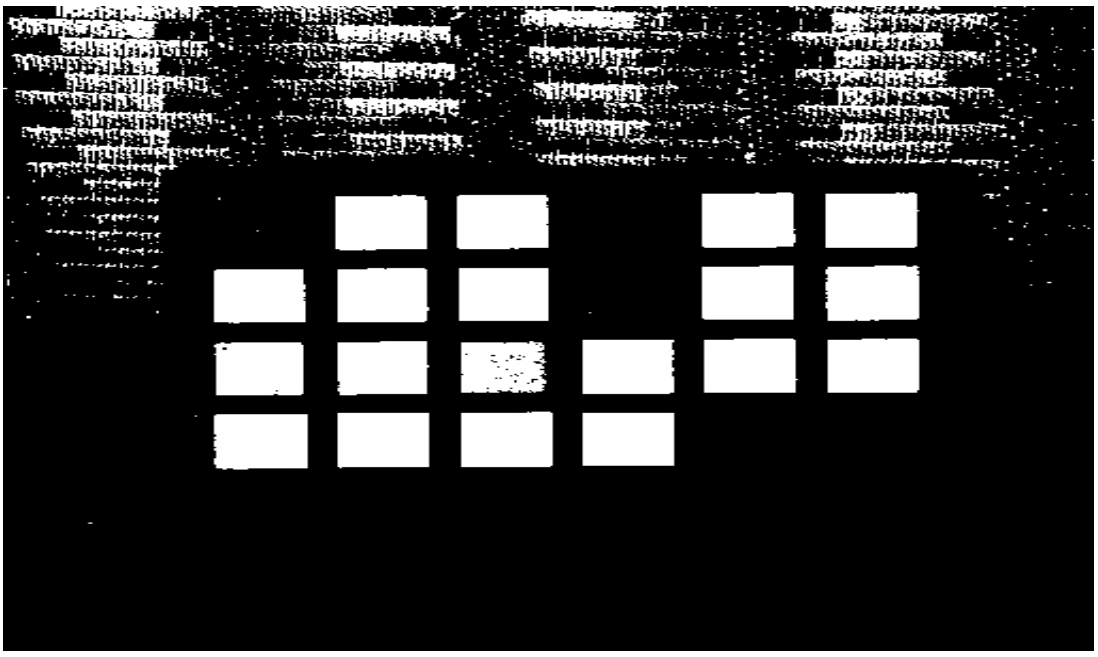


Image 3. Result of Fixed thresholding

(2) Random thresholding

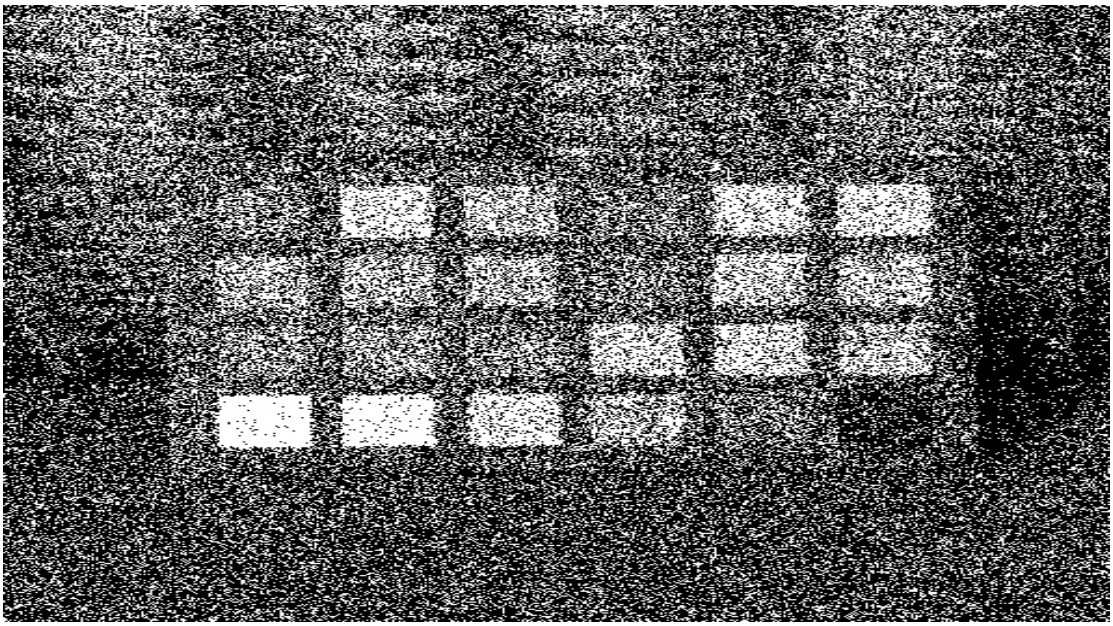


Image 4. Result of Random thresholding

(3) Dithering Matrix

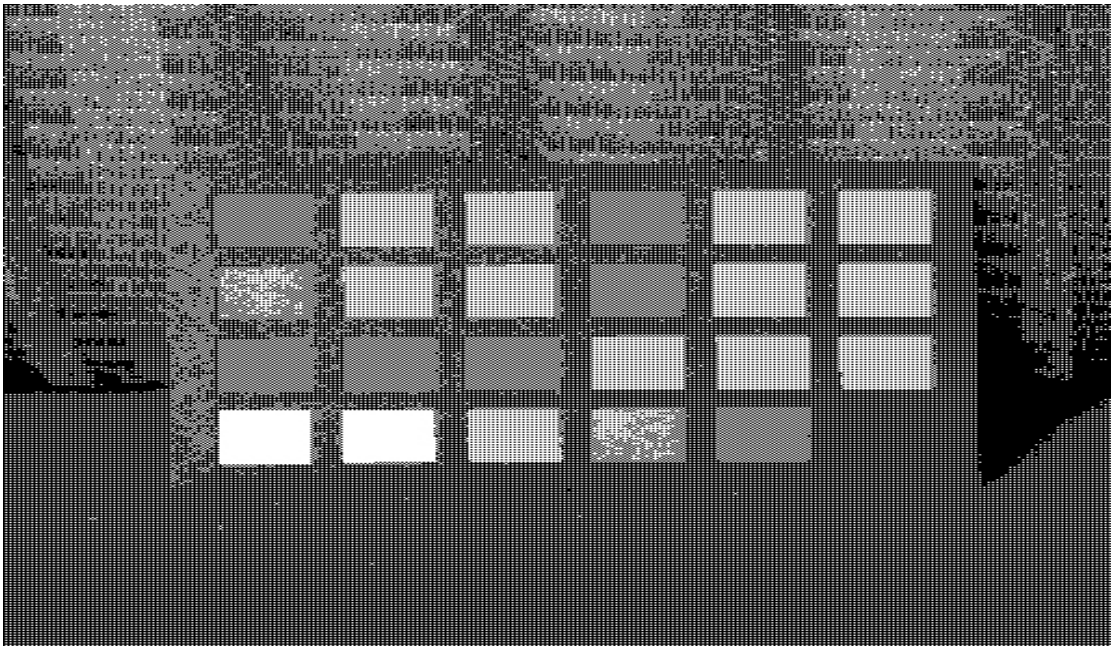


Image 5. Result of Dithering Matrix 2*2 size result image

Dithering Matrix 2*2 size

$$I_2 = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

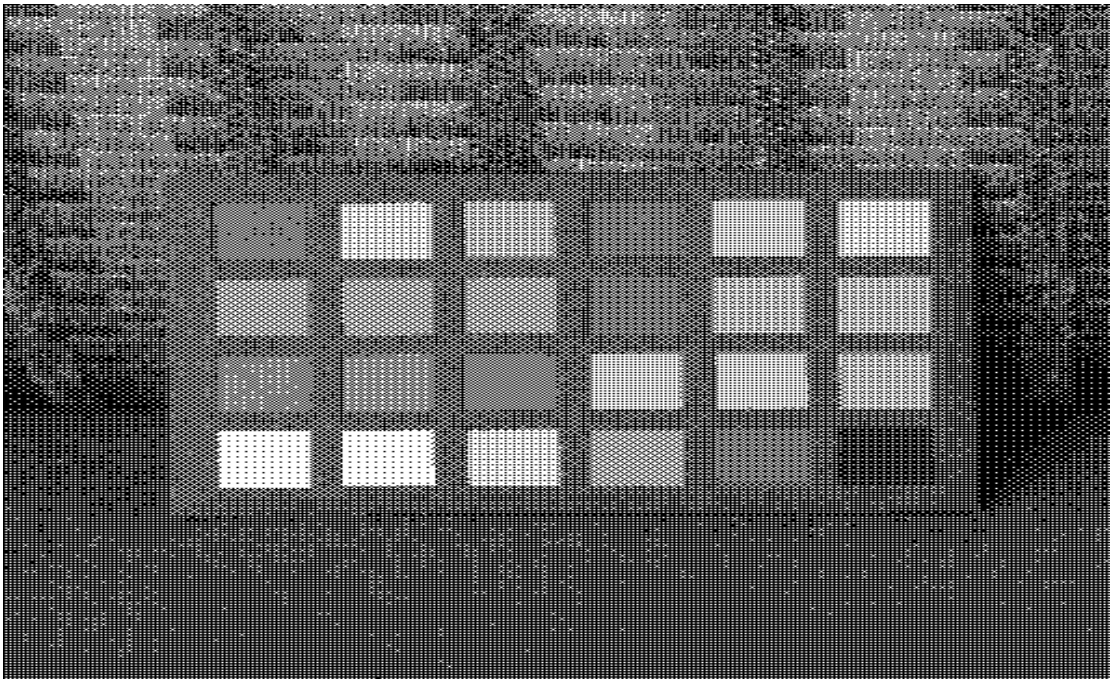


Image 6. Result of Dithering Matrix 4*4 size result image

Dithering Matrix 4*4 size

$$I_4 = \begin{bmatrix} 5 & 9 & 6 & 10 \\ 13 & 1 & 14 & 2 \\ 7 & 11 & 4 & 8 \\ 15 & 3 & 12 & 0 \end{bmatrix}$$

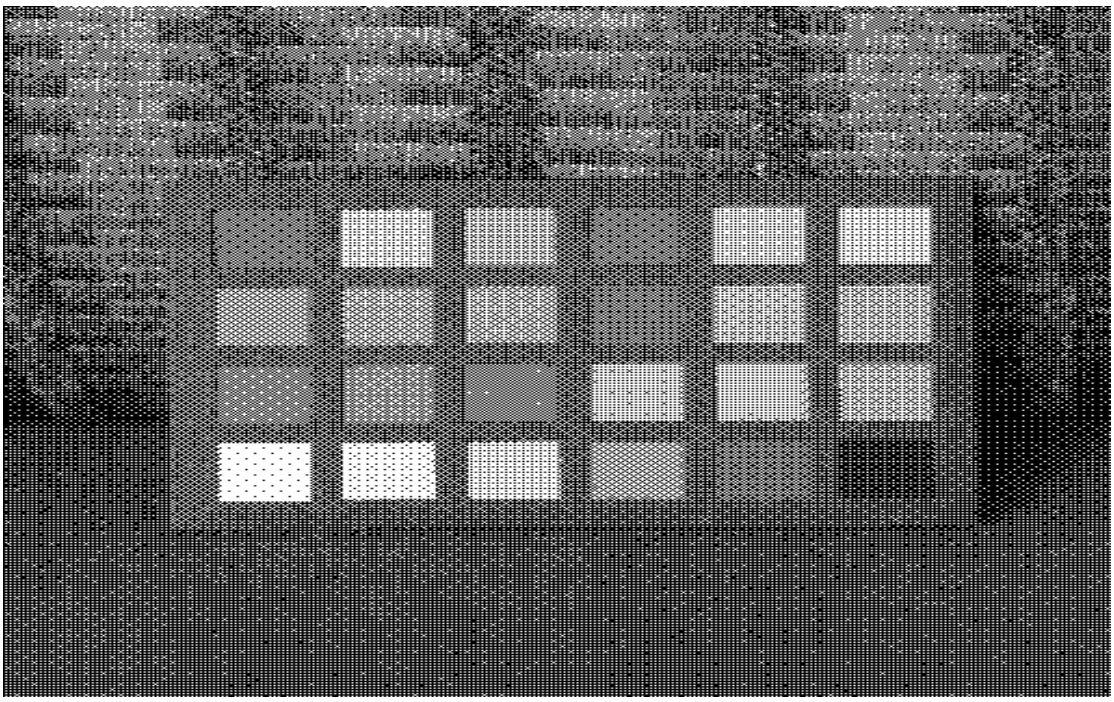


Image 7. Result of Dithering Matrix 8*8 size result image

Dithering Matrix 8*8 size

$$I_8 = \begin{bmatrix} 21 & 37 & 25 & 41 & 22 & 38 & 26 & 42 \\ 53 & 5 & 57 & 9 & 54 & 6 & 58 & 10 \\ 29 & 45 & 17 & 33 & 30 & 46 & 18 & 34 \\ 61 & 13 & 49 & 1 & 62 & 14 & 50 & 2 \\ 23 & 39 & 27 & 43 & 20 & 36 & 24 & 40 \\ 55 & 7 & 59 & 11 & 52 & 4 & 56 & 8 \\ 31 & 47 & 19 & 35 & 28 & 44 & 16 & 32 \\ 63 & 15 & 51 & 3 & 60 & 12 & 48 & 0 \end{bmatrix}$$

4.Discussion

(1). Comparing the results obtained by these algorithms in your report.

By comparing the results of these three methods and the five images, its obvious that the fixed thresholding has worst ability and the Dithering Matrix method has best result. To be specific, the Dithering Matrix method has better result when Dithering Matrix's scale equal to 8*8.

(2). If a screen can only display FOUR intensity levels, design a method to generate a display-ready Colorchecker image. Show your best result in gray-scale with four gray-levels (0, 85, 170, 255) and explain your design idea and detailed algorithm.

Description of my algorithm:

My algorithm combines random thresholding and fixed thresholding to get a better results as follow. My algorithm's steps as follow:

#1: Set the fixed threshold equals to 127, comparing the value of curate pixel with 127. Then set all pixels to two sets.

#2: In the two sub-sets, the pixels in the greater set compare with a random number that value between 128-255 and set the value to 170 or 255. Another set's pixels compare to a random number that value between 0-127 and set the value to 0 or 85.

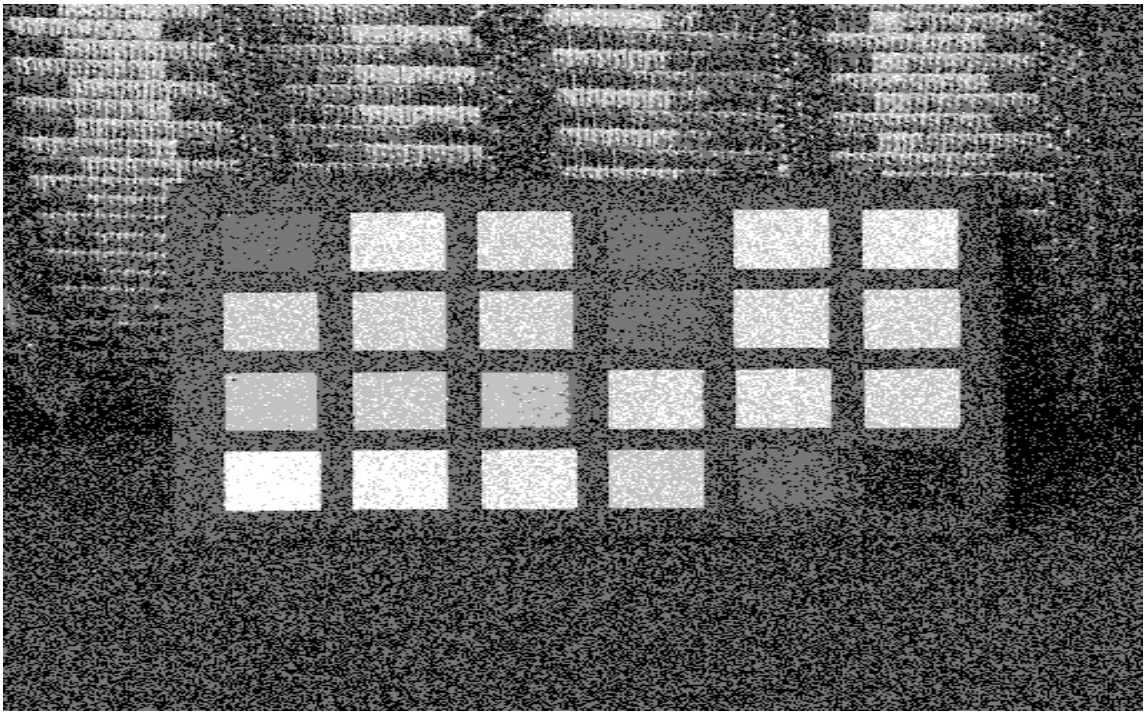


Image 8. Result of My algorithm of 4-values result image

Its obvious my algorithm has a good result due to I combined two different method. The fixed thresholding can clarify all pixels into two sets and not randomly set pixels value wrongly a lot. Secondly, the random method can help build up a good half-toning result in two sub-sets.

（b）Error Diffusion

1.Abstract and Motivation

If we just use the dithering method to get half-toning image, it is common to generate some error and decrease the visual effect of an image. Therefore, we can diffuse the error of current pixel to its surrounding pixels which are not processed by different ratio so as to make the color distribution more sparsely and disordered in the output image. The quantified error is the difference between input pixel and that of the value after comparison with threshold. Different method has different diffusion ratio to the its neighbors, there are three common used matrices as following.

2.Approach and Procedures

(1) Floyd-Steinberg's method

Scan the input image by serpentine routine, and the error diffusion matrix is:

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

We check pixels of the input image by ‘s’ shape using the method of fixed dithering threshold and calculate difference between input and output value and diffuse it to the current pixel’s neighbors by $\frac{7}{16}, \frac{3}{16}, \frac{5}{16}, \frac{1}{16}$ respectively based on their position.

(2) Jarvis, Judice, and Ninke (JJN) method

Do the same thing as (1), just change the multiplicative coefficient.

Error diffusion proposed by Jarvis, Judice, and Ninke (JJN), where the error diffusion matrix is:

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

(3) Stucki’s method

Error diffusion proposed by Stucki, where the error diffusion matrix is:

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

3.Experimental Results

(1) Floyd-Steinberg’s method

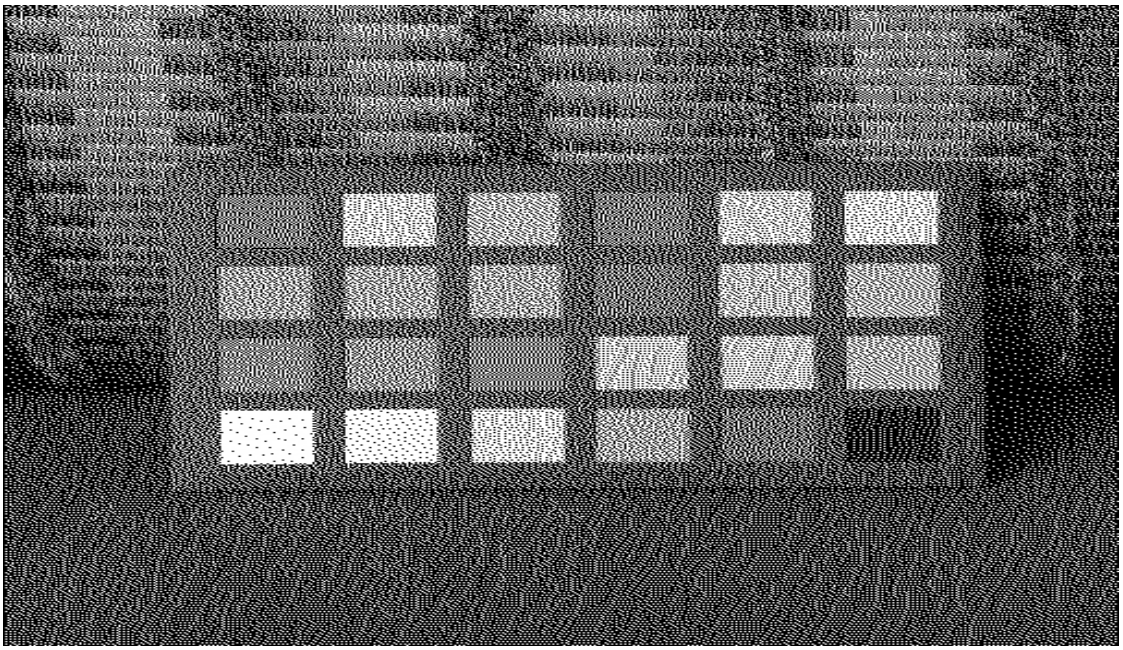


Image 9. Result of Floyd-Steinberg’s method

(2) Jarvis, Judice, and Ninke (JJN) method

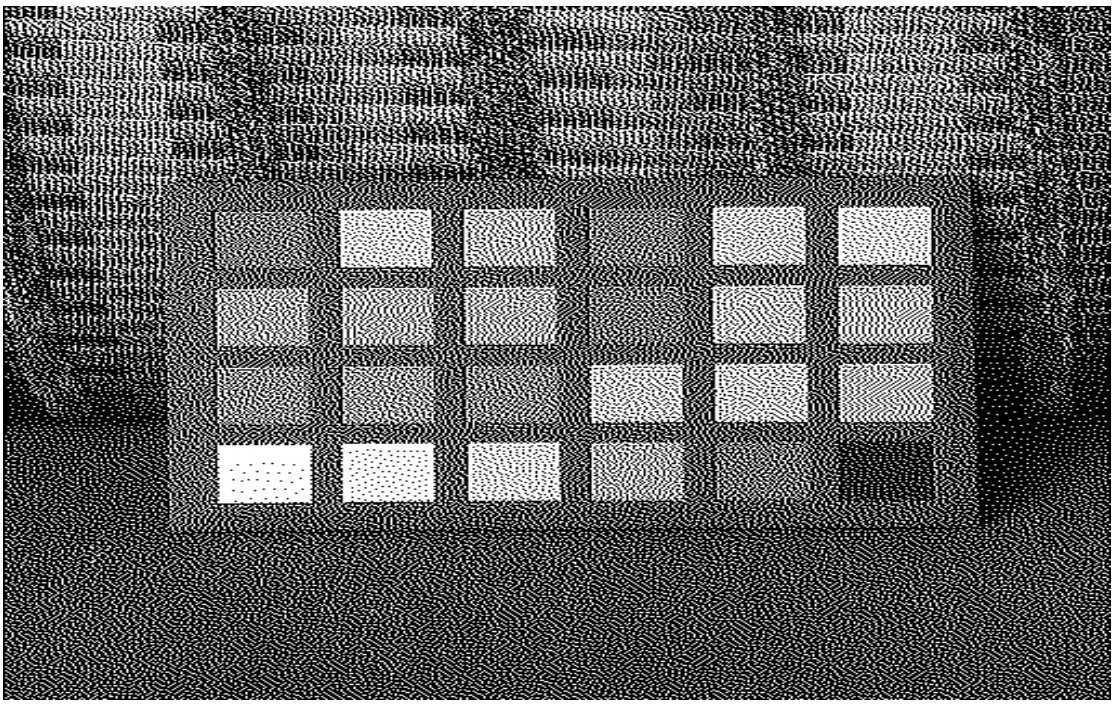


Image 10. Result of JJN method

(3) Stucki's method

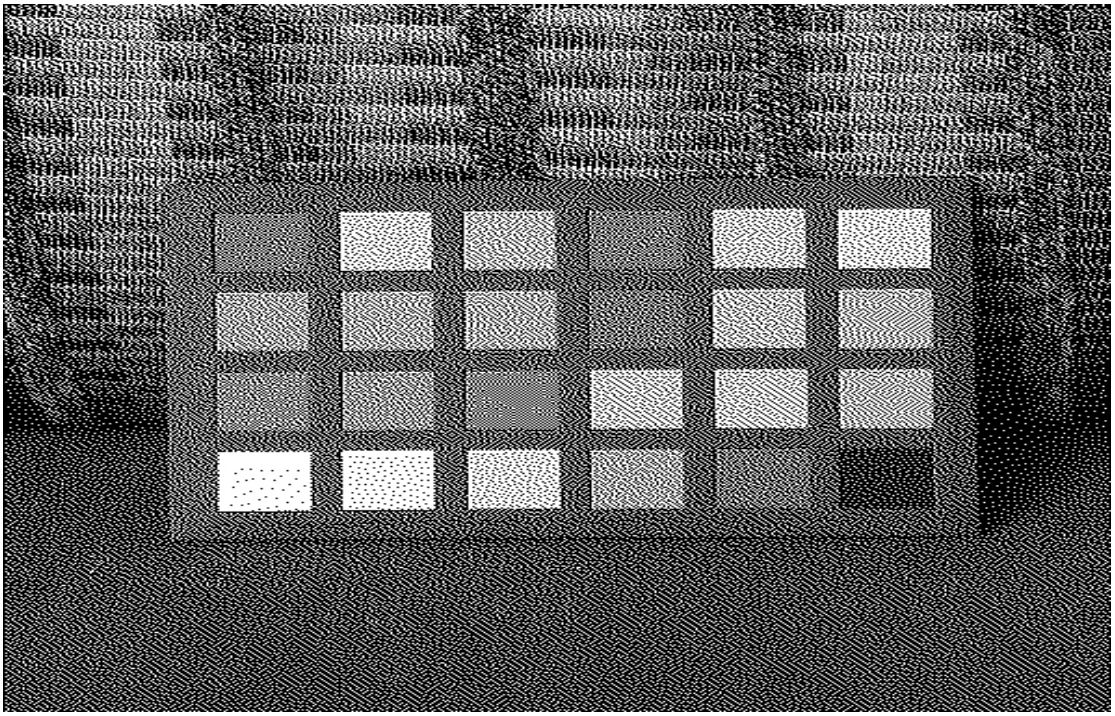


Image 11. Result of Stucki's method

4.Discussion

Compare the results obtained by these algorithms in your report.

According to the three methods I think Stucki's method has best result. Because this way's result has clarify edges in the image although this way is complexity with 、 matrix.

Describe your own idea to get better results. There is no need to implement it if you do not have time. However, please explain why your proposed method will lead to better results.
#1 idea: To get a better result, I think we can build a larger scale Error Diffusion matrix to get a result more randomly.
#2 idea: We can implement randomly method to do error diffusion

（c）Color Half-toning with Error Diffusion[1]

1.Abstract and Motivation

We also need to apply half-toning technique to color image. Since there are three channels RGB we need to deal with, it is more complicated than gray-scale image. Intuitively, we extend diffusion method to color image, so as to improve visual effect, we need to analyze and change the main color space, give a new definition to the distance measurement and similarity between colors based on colorful visual characteristics of human eyes.

2.Approach and Procedures

(1) Separable Error Diffusion

It is a simple method to process three color channels. We can apply the method we discussed above to these three channels separately. For example, we can use Floyd-Steinberg's error diffusion method to each channel and finally we will get the output color combination as following:

$$\mathrm{W} = (0,0,0), \mathrm{Y} = (0,0,1), \mathrm{C} = (0,1,0), \mathrm{M} = (1,0,0),$$

$$\mathrm{G} = (0,1,1), \mathrm{R} = (1,0,1), \mathrm{B} = (1,1,0), \mathrm{K} = (1,1,1)$$

Where the pixel value has been normalized.

(2) MBVQ- based error diffusion

The separable error diffusion method ignores that each color pixel has different brightness and it may change significantly among their neighbors. To obtain a better visual effect, we reclassify the color type based on brightness. One way is the MBVC method (Minimum Brightness Variation Criterion). It uses 4 basic colors which have the minimum brightness variation to fully represent any input color instead of 8 colors in the past. The four-color space is as follows: RGBK, WCMY, MYGC, RGMY, RGBM and CMGB. These combinations divide the RGB color cube into 6 different regions. Considering that if a color is in a tetrahedron which consist of four vertices in the RGB cube, these four vertices can totally represent the color.

The MBVQ- based error diffusion method is based on the new color space. Instead of setting a threshold for each pixel, we calculate the difference between the pixel value and 8 vertices of the color space cube, and then use the closest one as the output value. Finally, we can also apply Floyd-Steinberg's error diffusion method to distribute the quantized error to its neighbors so as to improve image quality.

3.Experimental Results

(1) Separable Error Diffusion

Image 12. Result of Separable Error Diffusion

(2) MBVQ- based error diffusion

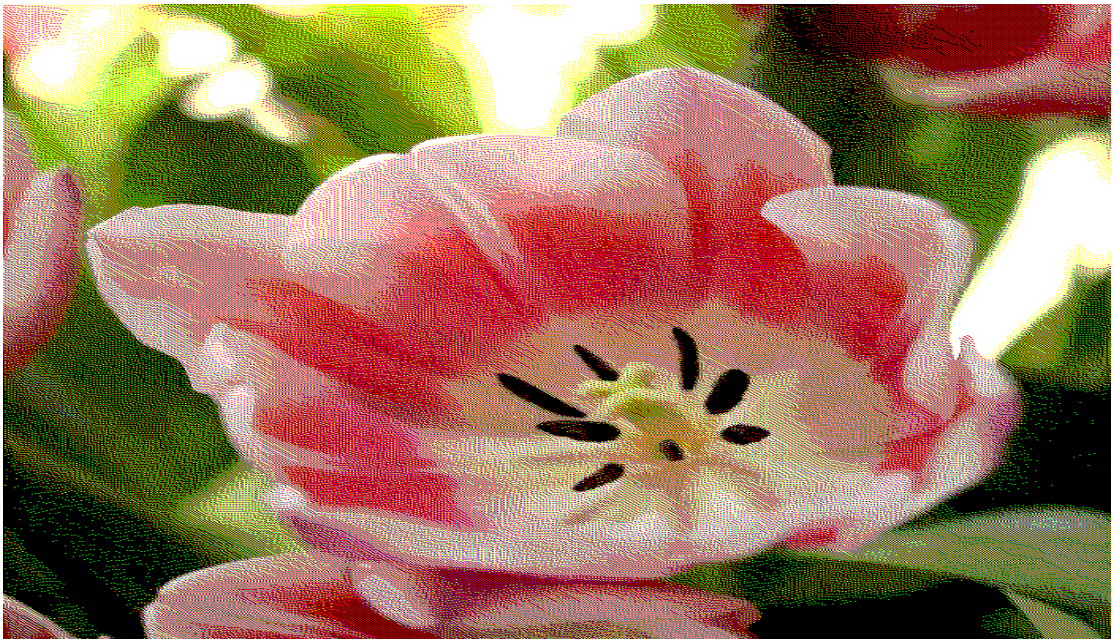


Image 13. Result of MBVQ- based error diffusion

4.Discussion

(1). Shortcoming of Separable Error Diffusion.

This way doesn't build up the relationship between three channels and easily result in different influence to different parts of the image.

(2). Compare the output with that obtained in Separable Error Diffusion and MBVQ-based error diffusion. Discuss the difference between these two methods.

The MBVQ based method has better result due to it considered the relationship between three channels of every pixel and the relationship between neighbor pixels by error diffusion. However, the separable error diffusion method only consider the relationship between neighbor pixels instead of different channels.

Problem 3: Morphological Processing

The morphology, i.e. mathematical morphology, is one of the most widely used technology in image processing. It mainly used to exact image information which are useful to describe regional shape, so that the following object identification can seize the essence of the most discriminating characters, such as the border and connectivity. We usually use gray value 0 (black) to represent object and use gray value 1 (white) to represent background after getting binary image of the input one. In digital image

processing, morphology is described with the aid of set theory. The structure element is used to deal with input binary image, like the mask. There are several morphological processing operations, including shrinking, thinning, skeletonizing. The main idea of them is to extract the important shape information of the object and remove their surroundings, the difference is the erosion degree. The effect of morphological operation depends on the size and content of the structural elements and the nature of the logical operation.

The hit-or-miss transformation is the basic tool for morphological shape detection, and is a morphological operator used to find the local pattern of pixels by hot-or-miss filter. It is called template matching procedure actually. The filter size is always 3*3, we denote surrounding pixels as from X_0 through X_7. When the binary image has exact same element with the filter mask, we call it 'hit' and change the center value, 0->1or 1->0, if there is one element, we call it 'miss' and remain the center pixel value. When the size is not big enough, we apply it twice to find the potential hit point and remove it in the second stage.

(a) Shrinking

1.Abstract and Motivation

Shrinking is used to get one point like the geometric center of the original binary image of the original object which is the most aggressive one.

2.Approach and Procedures

My algorithm's steps as follow:

The logical equation as follow: $G(i, j) = F(i, j) \cap [M(i, j) \cup P(M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7)]$

#1: Build up binary image **F** of the input grayscale image.

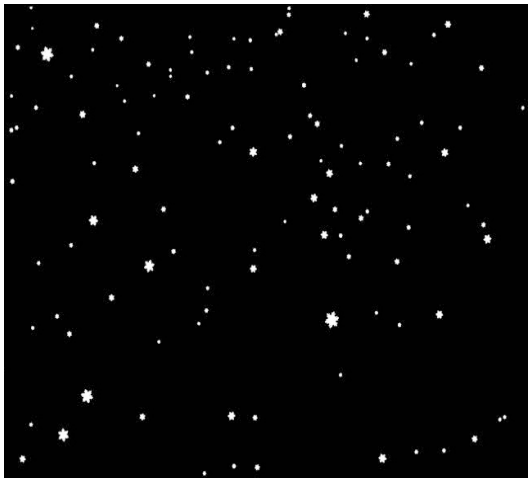
#2: Using Shrink pattern to handle **F** and get the result as **M**.

#3: Using Unconditional Shrink pattern to handle **M** and get the result as **G**, also set the **G**'s values to the image **F**.

#4: Keep on do steps 2 and 3 until there are no more change to **G**.

#5: Counting the number of pixels that grayscale equal to 255 and output the number.

3.Experimental Results

	
The original stars image	**The output counting image**

Sheet 2. Star result

The output that includes the number of stars and the number of star sizes.

```
/Users/hangdong/CLionProjects/ee569/cmake-build-debug/ee569 stars.raw stars_out.raw 480 640
readraw IO is working
the number of stars is:112
the number of stars size is:12

Process finished with exit code 0
```

4.Discussion

According to the result that by shrinking method every star leave a pixel that equals to 255.

(b) Thinning

1.Abstract and Motivation

Thinning is an operation of morphological to get a single line of the original object which has equidistance to its two edges.

2.Approach and Procedures

My algorithm's steps as follow:

The logical equation as follow: $G(i, j) = F(i, j) \cap [M(i, j) \cup P(M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7)]$

#1: Build up binary image **F** of the input grayscale image.

#2: Using Thinning pattern to handle **F** and get the result as **M**.

#3: Using Unconditional Thinning pattern to handle **M** and get the result as **G**, also set the **G**'s values to the image **F**.

#4: Keep on do steps 2 and 3 until there are no more change to **G**.

#5: Counting the number of pixels that grayscale equal to 255 and output the number.

3.Experimental Results

	
The original jigsaw_1 image	**The output main structure image**

Sheet 3. Jigsaw result

4.Discussion

By this way I can get the main structure(skeletonizing) of the input jigsaw_1.raw. What is more, every pixel keep connectivity with each other.

(c) Skeletonizing

1.Abstract and Motivation

Skeletonizing is some like thinning but it largely keeps internal information and connectivity of the original binary object by removing foreground pixels in a binary image .Its skeleton of median axis extends to the edge of image corner.

2.Approach and Procedures

My algorithm's steps as follow:

The logical equation as follow: $G(i, j) = F(i, j) \cap [M(i, j) \cup P(M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7)]$

#1: Build up binary image **F** of the input grayscale image.

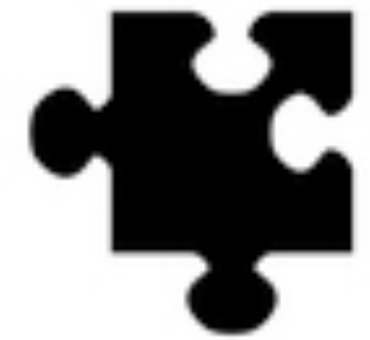
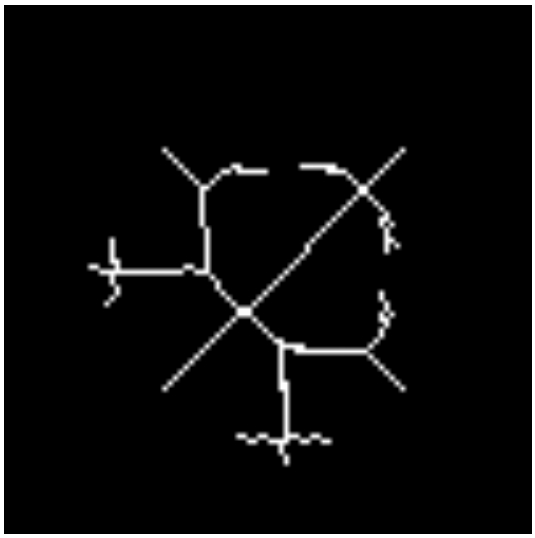
#2: Using Skeletonizing g pattern to handle **F** and get the result as **M**.

#3: Using Unconditional Skeletonizing pattern to handle **M** and get the result as **G**, also set the **G**'s values to the image **F**.

#4: Keep on do steps 2 and 3 until there are no more change to **G**.

#5: Counting the number of pixels that grayscale equal to 255 and output the number.

3.Experimental Results

	
The original jigsaw_2 image	**The output main structure image**

Sheet 4. Jigsaw_2 result

4.Discussion

By this way I can get the main structure of the input jigsaw_2.raw. The result maintains the skeletonizing of jigsaw. What is more, every pixel keep connectivity with each other.

(d) Counting game

1.Abstract and Motivation

To count the number of pieces in the board image board.raw, I used the connecting analyze method. The connecting analyzing method marks neighbor pixels with same value and after iterations in different order to get the number of pieces.

2.Approach and Procedures

My steps to count the number of pieces as follow:

#1: Build up binary image and reverse the pixels values.

#2: Set X equals to 1.

#3: Start traversal in order of rows to decide the neighbor pixels' value, if the curate pixel equals to 255 and no neighbor pixels equals to 255, set the curate pixel equal to X and X++.

#4: Iteration #2 in different order.

#5: Count different number in the image and get the number of pieces Y.

#6: Split the original image into Y parts and only keep the edge point which only has one neighbor pixel not equals to 0 in each sub-image. And mark the images have same edge point to same set.

3.Experimental Results

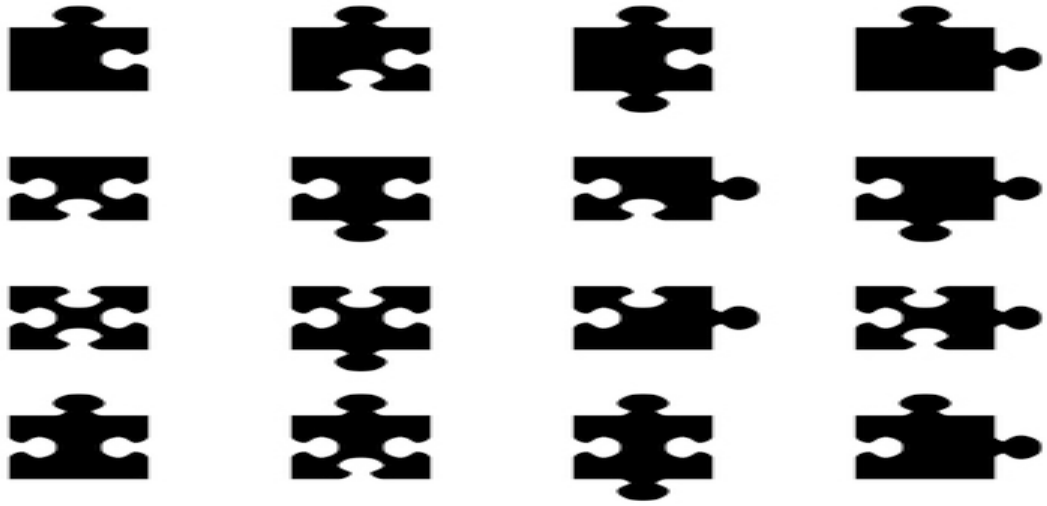


Image 14. The original image of boad.raw

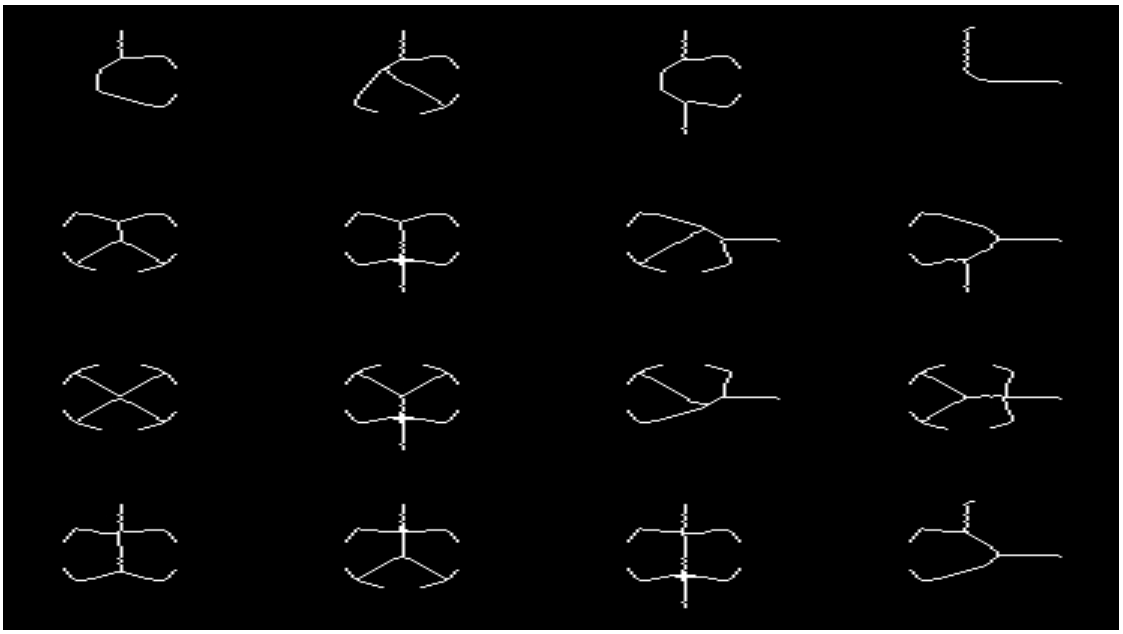


Image 15. The skeletonizing structure of the board.raw

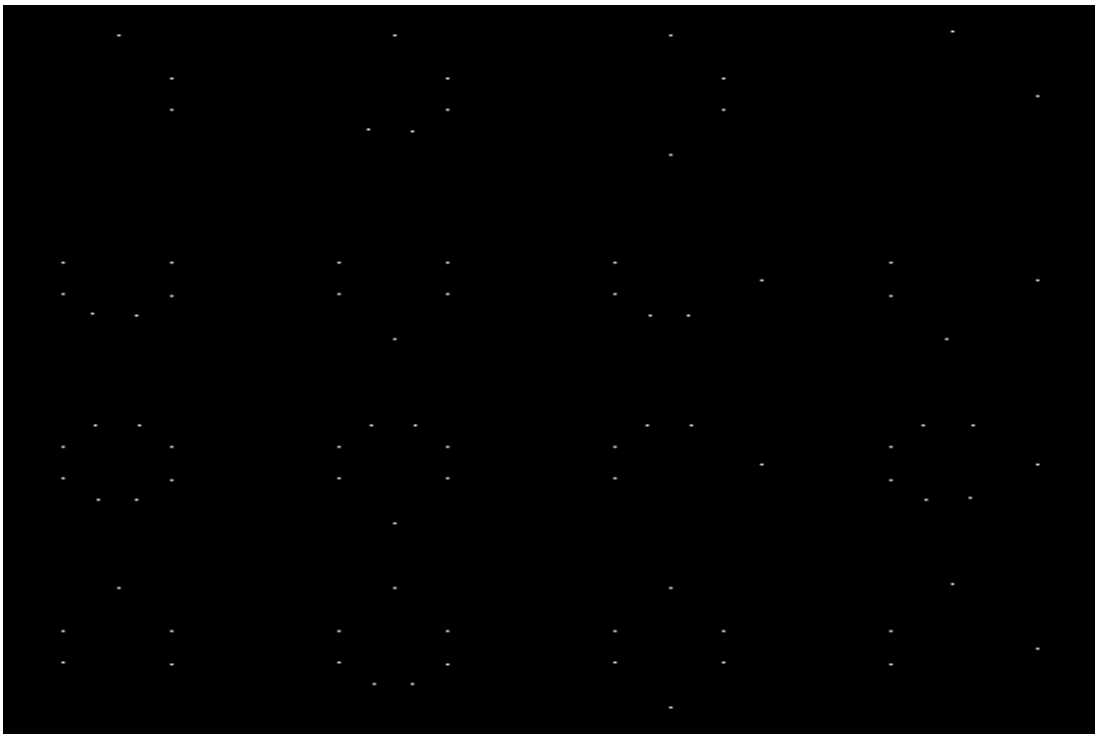


Image 16. The edge pixels of all pieces in board.raw

Assuming the pieces ID as follow order :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

The edge pixels number in each piece as follow:

3	5	4	2
6	5	5	4
8	7	5	7
5	7	6	4

Based on the edge pixels image, its obvious that we can set 4 sub-sets:

{2,6,7,11,13},{3,8,16},{10,12,14},{5,15}

Then by comparing the spatial relationship of edge pixels in each sub-set, finally I decide the final sets as follow: (ID of pieces)

{1},{2,7,11},{3},{4},{5},{6,13},{8,16},{9},{10,12,14},{15}

Sheet 5. Counting game result

4.Discussion

In my algorithm I just clarify the different pieces into few sets based on the edge pixels number and their spatial relationship. It can not efficiently clarify the pieces that include same edge pixels and similar spatial relationships.

Reference

[1] D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones",
HP Labs Technical Report, HPL-96-128R1, 1996.