Qidi Han (6814891757)
EE-569
Homework #2
Feb-16-2020

Problem 1: Image Demosaicing

I. Abstract and Motivation

In the recent year, the technology of edge detection had become more and more popular. The technology is using widely in our daily life. Many big companies make those technology in their products. There are many ways to do the edge detection. For general way of edge detection is detecting the sharp intensity change between each pixel. Edge detection extract the important parts of the image. Different edge detection will give us different preform. To knowing those algorithms helped us to better understanding their technology and using those technology will help us to perform a better evaluation on images. In this project, we are going to learn three different technology relate to the edge detection. The first call Sobel edge detector. The second called canny edge detector. The last one called structured edge detector.

II. Approach and Procedures

a. Sobel Edge Detector

There are two images that need to do the Sobel edge detector. There are gallery and dogs image. The first step of this experiment is converting the RGB images to grayscale image. The formula below is converting RGB image to grayscale image.

$$Y = 0.2989 * R + 0.5870 * G + 0.114 * B \quad (1)$$

The second step is calculating the probability map for each image, the probability map also called normalized gradient magnitude map. Before doing the magnitude map, I have to compute the x-gradient and y-gradient value for each picture. Do the same convolution process with last assignment. For x-gradient, using filter [- 1 0 +1; -2 0 2; -1 0 1]. For y-gradient, using filter [1 2 1; 0 0 0; -1 -2 -1]. After, we do the convolution, we need to normalize each x-gradient and y-gradient values to 0-1 respectively. Figure 1 shows the x-gradient and y-gradient for Gallery image. Figure 2 shows the x-gradient and y-gradient for Dogs images. My normalization method is adding the x-gradient and y-gradient with minimal number of the matrix and divide by maximum number of the matrix. After those steps. It will give us a x-gradient and y-gradient value with 0-1 respectively. Finally, use the equation below to combine the x-gradient and y-gradient to magnitude map. Figure 3 shows the normalized gradient magnitude map for Gallery image. Figure 3 shows the normalized gradient magnitude map for Dogs image.

$$|\nabla f| = \sqrt{gx^2 + gy^2} \quad (2)$$

The next step is setting a threshold to make the probability map into a binary map. Using the formula below to determine the binary number. Figure 5a,5b,5c shows binary image with three different thresholds for Gallery image. Figure 6a,6b,6c shows binary image with three different thresholds for Dogs image.

$$output(i,j) = \begin{cases} 255 & input(i,j) > threshold \\ 0 & otherwise \end{cases} \quad (3)$$

b. Canny Edge Detector

For the canny edge detector, I'm going to explain the Non-maximum suppression first in the discussion, then I will try some different high and low threshold value to find the best solution for the result. The canny edge detector is very useful edge detection before the machine-learning base detection. The first step of the canny edge detector is using the Gaussian filter to remove the noise. Then calculate the gradient magnitude map, finally use the non-maximum suppression method. There are two threshold values for Canny edge detector. The high and low threshold. The pixel that lower than low threshold will consider as not edges. The pixel that higher than high threshold will consider as edges pixels, and the pixels that between two thresholds will consider as edges if the neighbor pixels is the edge pixel. Canny edge detector is much better detector than Sobel edge detection. In our project, I will use `edge(grey,'Canny')` function to the detection. This is one of the functions in the MATLAB image processing toolbox. Figure 7a, 7b, 7c and 8a, 8b, 8c shows the result for Canny edge detections for both Gallery and Dogs images.

c. Structured Edge Detector

Structured edge detection is the one of the best edge detection method for edge detection. As I mentioned before, Canny edge detection is a useful edge detection before the machine-learning base detection. The first big different between structured edge detector with Canny edge detector is the structured edge detection can input a colorful image. but the Canny edge detection, we have to convert the image to greyscale. Structured edge detection is the machine-learning base detection. For machine-learning base detection, there are three parts of the process. There are label the class, training the training data, and test the input image. In this problem, I will make a flow chart and explain the flow chart in my discussion part. The main algorithm of Structured edge detection is the random forest classifier. I will explain the random forest classifier in the discussion part, I will combine the flow chart and random forest to explain the whole process. The final step is applying the Structured edge detector to Gallery and Dogs image. I will choose the best parameters to make a best solution. Then I will compare the visual result with Canny detector and the Structured edge detector. Figure 9 shows the structured edge detector for Gallery image. Figure 10 shows the structured edge detector for Dogs image. I will repeat the experiment with different edge detector methods.

d. Evaluate the performance

There are five different type of ground truth image that proved by the instructor for both Gallery and Dogs image. In this problem, we are going to calculate the precision and recall for each ground truth. The formula for precision is below

$$Precision\ P = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive} \quad (4)$$

The formula for recall is below

$$Recall\ P = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative} \quad (5)$$

After calculating the precision and recall, I will compute the F score. The higher F score will have a better edge detector. The formula below is for calculating the F score.

$$F = \frac{2 * P * R}{P + R} = \frac{2}{1/P + 1/R} \quad (6)$$

For each ground truth, we also need to compute the mean precision and the mean recall. And calculate the F score for the mean precision and mean recall. And for each threshold, I will compute the F score with all five pictures. And I will plot the data in the Figure 11. Figure 11a,11d shows the plot for Sobel edge detection. Figure 11b,11e shows the plot for Canny edge detection. Figure 11c,11f shows the plot for Structured edge detection. and Table 1 Table 2 to shows the result the Gallery image and Dogs image. Finally, I will discuss the result in the discussion part. Table 3a and 3b show the best F score for each edge detection. Table 4 will show the performance of each different edge detectors.

III. Experimental Results

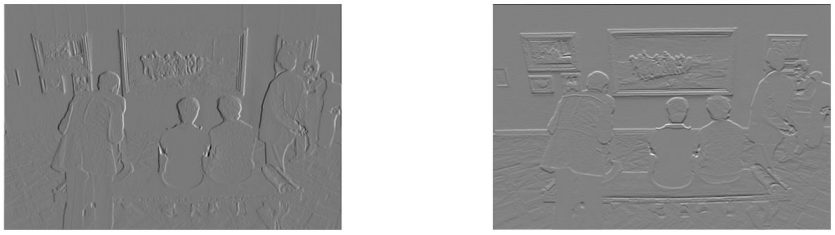


Figure 1 shows the x-gradient and y-gradient for Gallery image

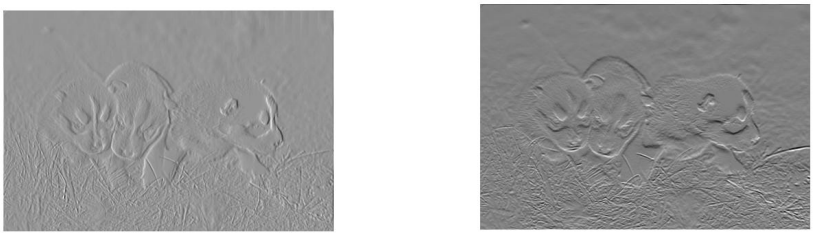


Figure 2 shows the x-gradient and y-gradient for Dogs image

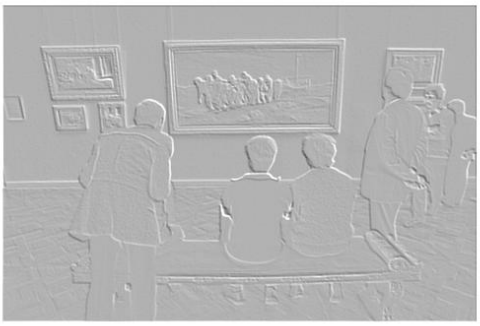


Figure 3 shows the normalized gradient magnitude map for Gallery image

Figure 4 shows the normalized gradient magnitude map for Dogs image

Figure 5a shows binary image with 68% thresholds.

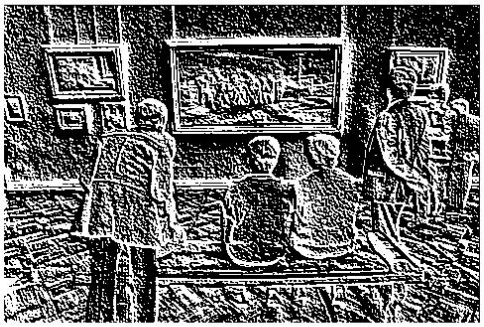


Figure 5b shows binary image with 70% thresholds.

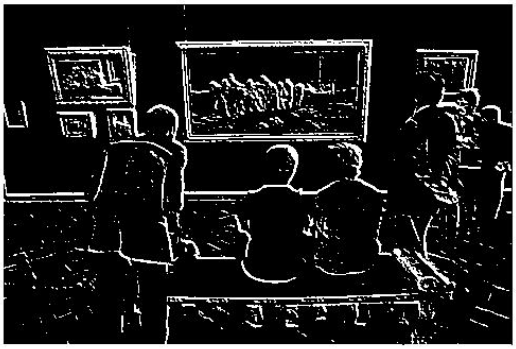


Figure 5c shows binary image with 73% thresholds.

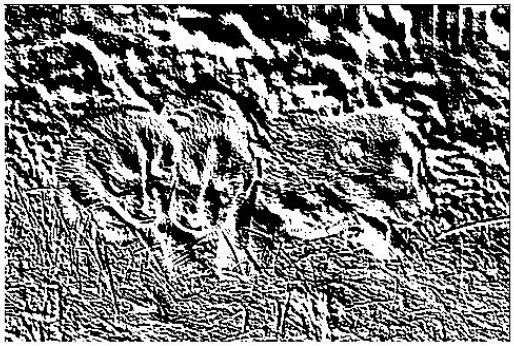


Figure 6a shows binary image with 77% thresholds.

Figure 6b shows binary image with 80% thresholds.

Figure 6c shows binary image with 85% thresholds.

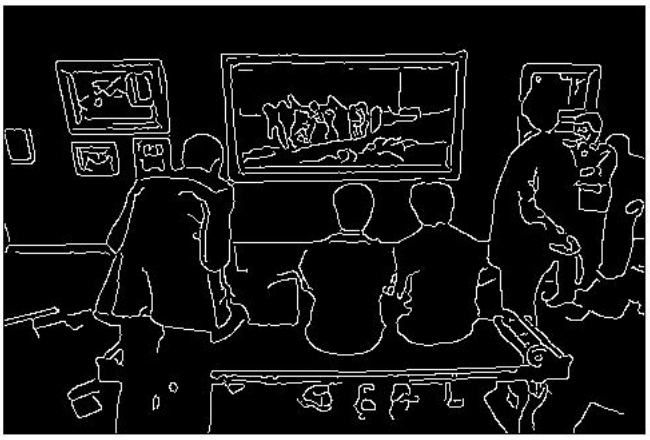


Figure 7a. shows the result for Canny edge detections for Gallery with threshold `[0.05 0.15]`

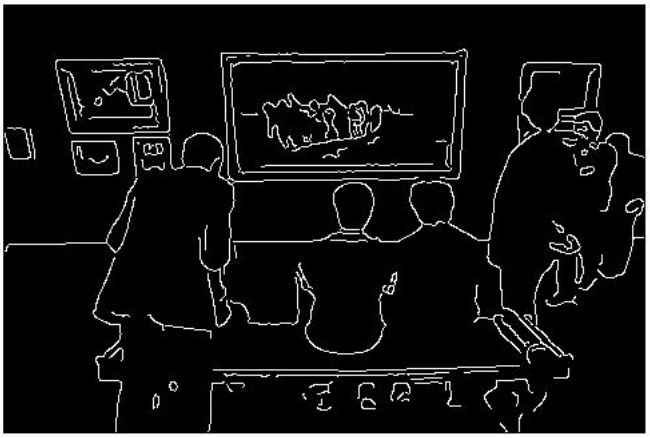


Figure 7b. shows the result for Canny edge detections for Gallery with threshold `[0.15 0.20]`

Figure 7c. shows the result for Canny edge detections for Gallery with threshold `[0.20 0.25]`

Figure 8a. shows the result for Canny edge detections for Dogs with threshold `[0.15 0.20]`

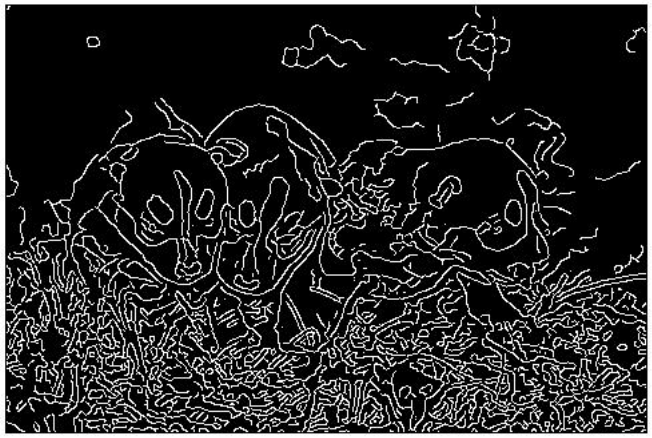


Figure 8b. shows the result for Canny edge detections for Dogs with threshold `[0.15 0.20]`

Figure 8c. shows the result for Canny edge detections for Dogs with threshold `[0.15 0.20]`

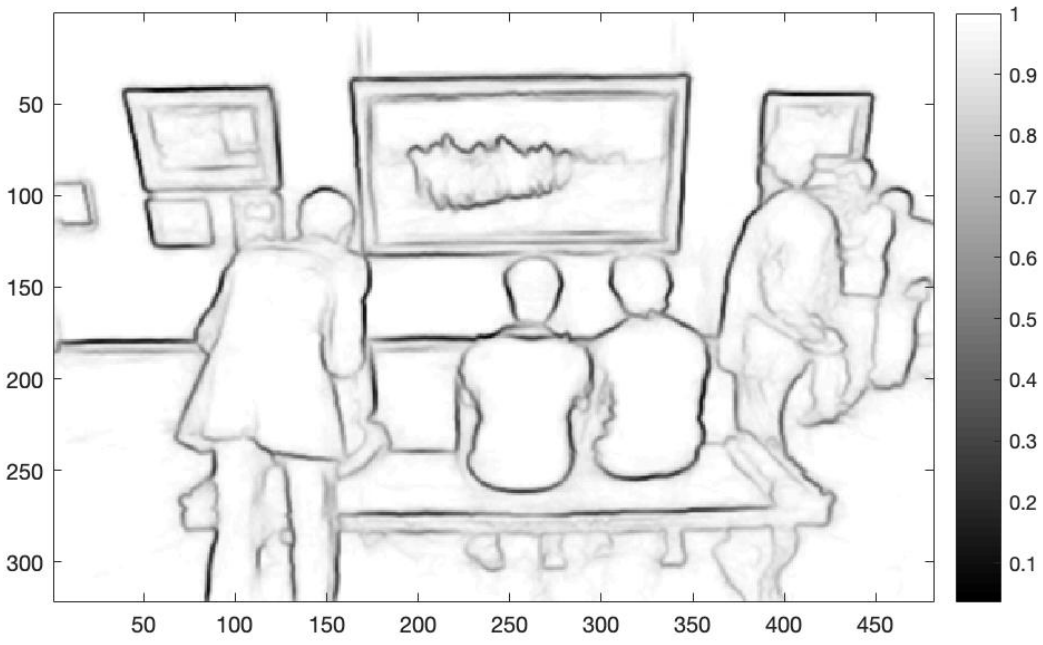


Figure 9 shows the structured edge detector for Gallery image.

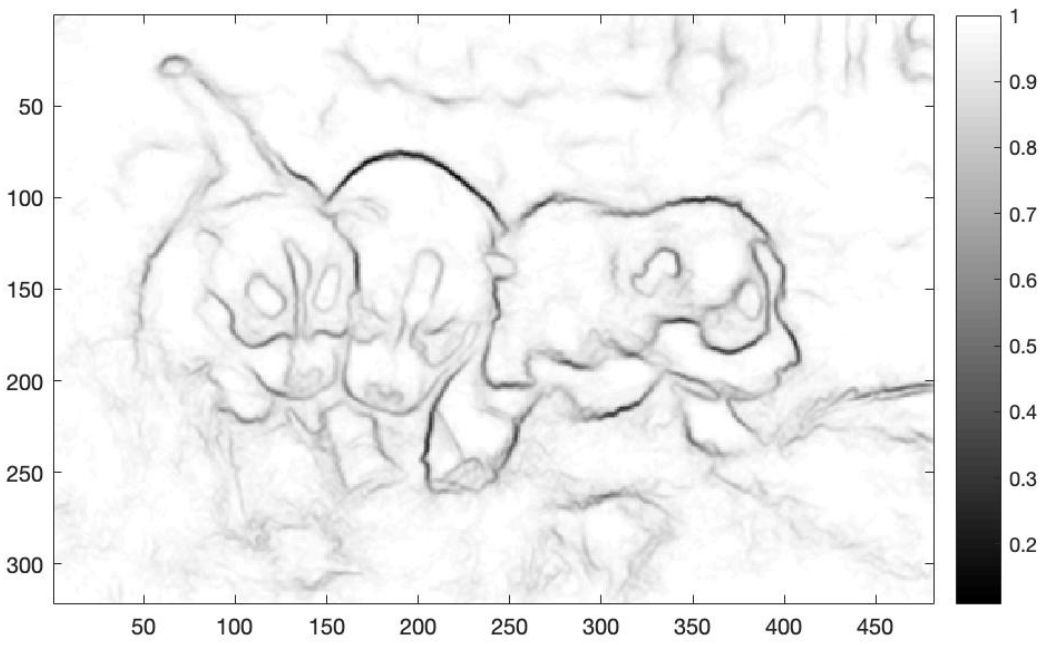


Figure 10 shows the structured edge detector for Dogs image.

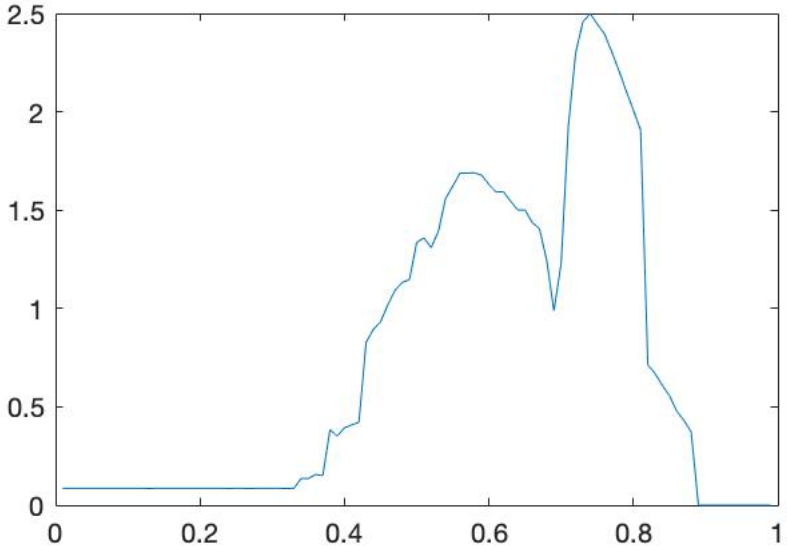


Figure 11a shows the plot for Sobel edge detection for Gallery image.

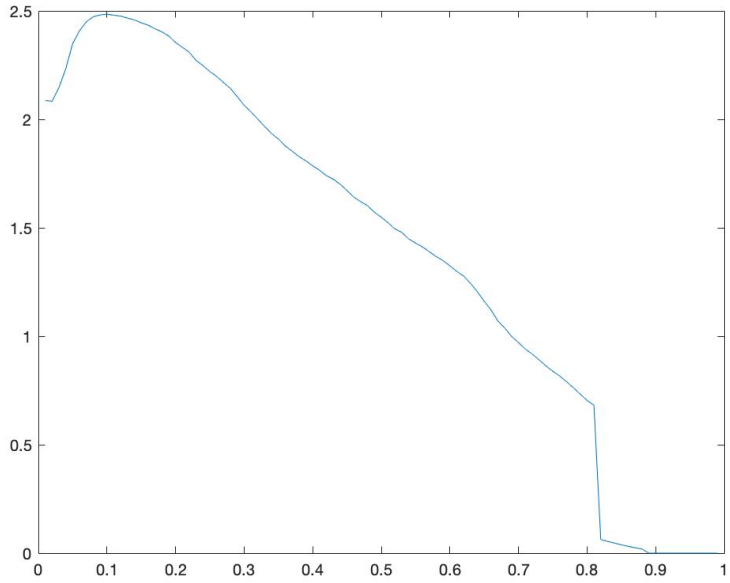


Figure 11b shows the plot for Canny edge detection for Gallery image.

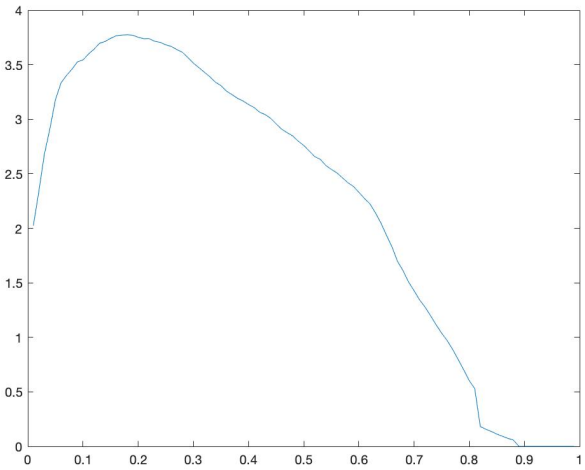


Figure 11c shows the plot for Structured edge detection for Gallery image.

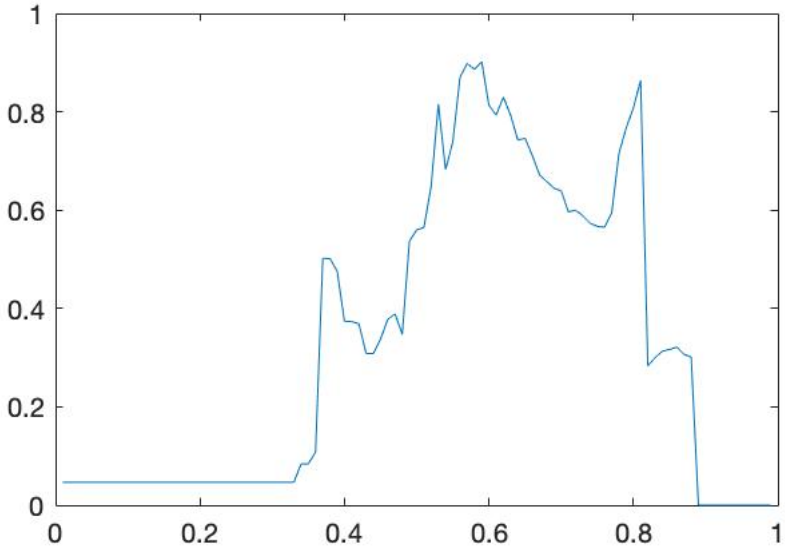


Figure 11d shows the plot for Sobel edge detection for Dogs image.

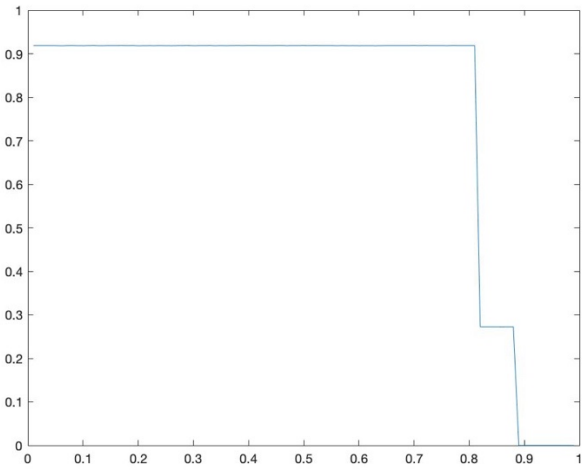


Figure 11e shows the plot for Canny edge detection for Dogs image.

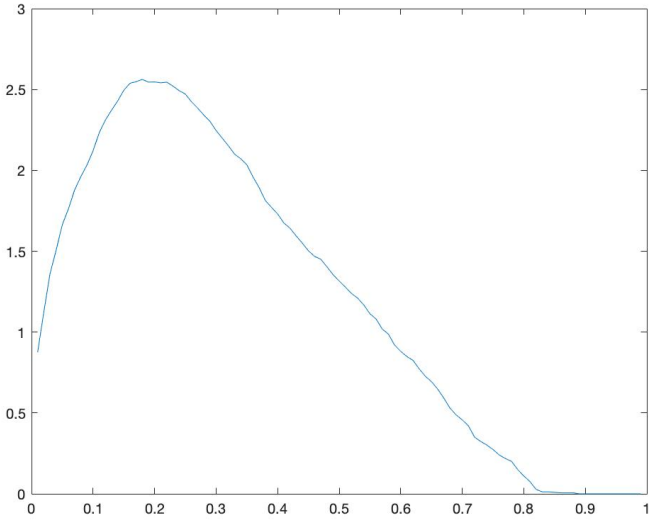


Figure 11f shows the plot for Structured edge detection for Dogs image.

Table 1 shows mean precision, mean recall and mean F for Gallery image.

Sobel Edge	GT1	GT2	GT3	GT4	GT5
Mean Recall	0.213482229	0.22115468	0.21135248	0.22310214	0.22312529
Mean Precision	0.252118039	0.24415631	0.24232015	0.24541916	0.24540566
Mean F	0.139708231	0.13640504	0.13477337	0.15702422	0.15704828
Canny Edge					
Mean Recall	0.976506588	0.97955397	0.96831468	0.96473137	0.96477768
Mean Precision	0.326929107	0.15127679	0.15329562	0.1503951	0.15038826
Mean F	0.437180853	0.24469901	0.24539818	0.24334252	0.24333308
SE Edge					
Mean Recall	0.431020286	0.45049895	0.42877129	0.45334651	0.45332592
Mean Precision	0.708119386	0.6825865	0.64753536	0.64243172	0.64241678
Mean F	0.449684202	0.45119842	0.43504602	0.45148494	0.45146052

Table 2 shows mean precision, mean recall and mean F for Dogs image.

Sobel Edge	GT1	GT2	GT3	GT4	GT5
Mean Recall	0.237184507	0.1846535	0.2262707	0.22880578	0.22880578
Mean Precision	0.092050373	0.11803101	0.10500649	0.04414326	0.044144
Mean F	0.061847655	0.07384791	0.07048282	0.05855749	0.05855856
Canny Edge					
Mean Recall	0.93519171	0.80593627	0.8989092	0.92362765	0.92358096
Mean Precision	0.078931373	0.1413621	0.11042214	0.06536691	0.06536258
Mean F	0.132704559	0.21734455	0.17859888	0.12109275	0.12108472
SE Edge					
Mean Recall	0.308576723	0.18642125	0.22843616	0.32075765	0.32081991
Mean Precision	0.619581457	0.6475701	0.62659153	0.58203681	0.58203045
Mean F	0.267101135	0.19882776	0.22345774	0.2615642	0.26158121

Table 3a Best F for Gallery image

	Sobel	Canny	SE
Thresholds(%)	0.75	0.11	0.19
Best mean F	2.498979221	2.484856919	3.773569087

Table 3b Best F for Dogs image

	Sobel	Canny	SE
Thresholds(%)	0.60	0.09	0.19
Best mean F	0.901880465	0.918758582	2.562411175

Table 3 Best F for each edge detection

	Sobel	Canny	SE
Pros	1. Fast to process with the images.	1. non-maximal suppression	1. machine-learning base

		2. can deal with some noise 3. open source 4. easy to use	2. strong edge (clean edges) 3. open source 4. can deal with noise
Cons	1. Not machine-learning base 2. Blur the edge sometimes (weak edges) 3. Cannot deal with noise 4. Need to try many different thresholds	1. Not machine-learning base 2. Week edges compare with structured edge detection.	1. Slowly processing 2. Need to train the data. (need more data for training)

IV. Discussion

1.a-1
Figure 1 and Figure 2 shows the normalized x-gradient and y-gradient for both images.
1.a-2
Use the formula 2, and Figure 3 and 4 shows the normalized gradient magnitude map.
1.a-3
Gallery image: Figure 5 shows the result. There are three different thresholds that the question used. There are 68%, 70% and 73%. From visual result perspective, the best threshold is 73%. Because 68% and 70% of the threshold have too much details, the edge is not smooth, there are many black and white dot in the non-edge area.
Dogs image: Figure 6 shows the result. There are three different thresholds that the question used. There are 77%, 80% and 85%. From visual result perspective, the best threshold is 80%. The 77% of thresholds have too much details. The 85% of thresholds have too less details. There are many black areas. We cannot determine the edge for 85% of thresholds.

1.b-1
Non-maximum suppression is filtering the pixels that not considering as edges. We only keep the local maximum value for the image. For the non-maximum suppression, we have to define the gradient angels. We use x-gradient and y-gradient to calculate the gradient angles. When we go through the image in very small filter, we will use this gradient to determine if we need to make the central point as 0 or remind the value. When the central point has largest number, then we will keep the same value for the central point, otherwise, we will make it as 0.

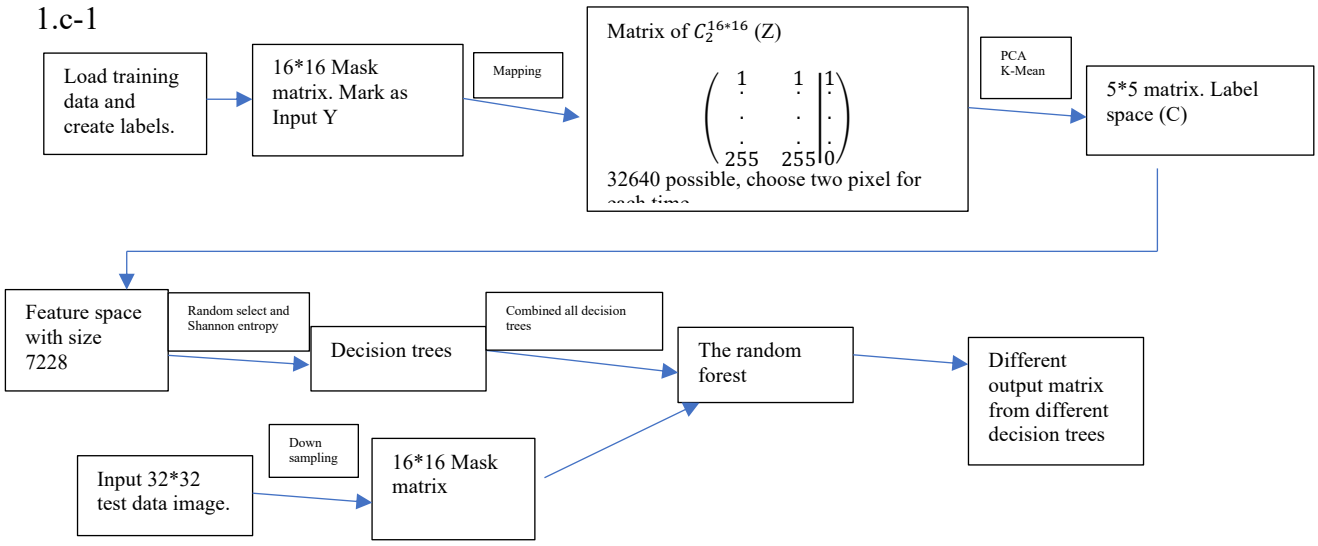
1.b-2
If the pixel value is higher than the high threshold, it will become the edge pixels. If the pixel value is lower than the low threshold, it will become the non-edge pixels which is rejected. When the pixel value is between the high and low threshold, we need to see if the neighbors

are edge pixels or not. If one of the neighbor pixels is edge pixel, we will make it as edge value. It also called connectivity.

1.b-3

For Gallery image, Figure 7 shows three different low and high thresholds. The three different thresholds are similar to each other's. But for range between [0.05 0.15] have the best performs from the visual perspective, because those edges give us a better result.
For Dogs image, Figure 8 shows three different low and high thresholds. I think the range between [0.15 0.2] has the best result. The Figure 8a has too much details. And Figure 8c has too less details. There is not enough edge pixel for Figure 8c.

1.c-1



The first step of this SE detection algorithm creating the label class. For the 16*16 mask matrix. we mark it ask input Y. Since the number of data in Y is too large, we have to make it smaller. Making Y into Z, we will compare each two point to construct a binary vector. 1 represent similar class, 0 represent different class. After we construct the matrix, but the data is still too big, we use PCA and k-mean to reduce the matrix to 5*5. And make those labels as label space. After we process the label, we are going to run the feature space.

For the feature space, we will use many different methods to separate the feature. For example, pixel lookups x(i, j, k) and pairwise differences x(i1, j1, k) − x(i2, j2, k). We are going to calculate number of features for each image. Since this is a colorful image, we have 3 color, 2 magnitude and 8 orientation channels for each image. It's total 13 channels for each image. We use a triangle filter with a radius of 2 to blur the image, then down-sampling 2 times, we get $32 \cdot 32 \cdot 13/4 = 3328$. Since our label resolution is 5*5 matrix. we have $C_2^{5*5} * 13 + 3328 = 7228$ features for each image. Then we random select feature to become our determination to build many decision trees. To build the random forest classifier, there are many decision trees in the process.

After, we build our decision tree, we are going to input our test data. The test data are 32*32, we have to down-sampling it to 16*16. Finally input the test data into our different decision tree and select the mode of those different result to our final result.

1.c-2
The process of decision tree construction is we use great entropy result to determine one feature as our first node. Then we use this feature to separate the data into server leaf node. Check if all the same class members classify to the same node, if not, we will use great entropy find the second feature to separate the data again. The second leaf node will generate server sub-nodes. Repeat the same process server time. Stop the process when all the same class members classify to the same node.
The principle of the RF classifier is combining all the tree independent prediction result to the final result by using the uniform distribution method. Make the process more accurate, because each decision have many bias from different features and training data.

1.c-3
Figure 9 and Figure 10 shows the structured edge detector result for Gallery and Dogs. The parameters that I choose is that maxDist=0.0075 as the default number, set the thrs number =99. Therefore, the threshold will run from 0.01 to 0.99 with 99 numbers. Those good number give me a better result. From the visual results, the SE detector gave us a better solution. It's obvious for Dogs image. For the Canny detector, the Dogs image will give us less or too much details result, but Structured edge detector, give us a very good result.

1.d-1
Figure 11, table 1, table 2, table 3 and table 4 shows the result for this question. From the best F score, we can make a conclusion that structured edge detection have the edge detection.
1.d-2
I think Gallery is easier to get high F measure, because I think the edge of gallery is clearer. The Dogs image shows many un-useful edges. The edge of Dogs is too complicated, and the edge is not obvious. We can see the same result from Canny edge detection.
1.d-3

- No, from the equation6. When precision is significantly higher than recall, if term (1/p)+(1/R) become smaller, the F will get big. The only way to make the (1/p)+(1/R) smaller is to make the P as greatest as they can, and also make the R is big number too. But this is not possible. It's conflict with our statement. We cannot increase both number at one time.
- I will use method of using derivative to find the maximum. Set P+R=c. F=2PR/(P+c). Insert P+R=c in to the equation. We got F=2P(c-P)/c. Find dF/dP =2-4P/c. To find the local maximum, we set dF/dP=0; we got 2-4P/c=0, then we have c=2P. Finally, put c=2P back to the equation of P+R=c. We got R=P. When R=p, we will have maximum number F. F measure reaches the maximum when precision is equal to recall.

Problem 2: Digitial Half-toning

I. Abstract and Motivation

Half-toning is a method that using in the tradition printing device. It creates the continues tone for the images. It also calls screening method. There are many algorithms that using for

half-toning. Different algorithms will have different method to place the black dots. The reason that we are using the half-toning, it's because for the greyscale image, there are 256 colors, and for the colorful image, there are 16.7 million colors. For the printing device, we don't have those many shadows of grey and wide image colors. Therefore, half-toning is necessary method for those print devices. In our project, we will use many different algorithms to achieve the digital half-toning. The first few pictures are grey-scale image, the last picture is colorful image.

II. Approach and Procedures

There are three parts of this experiment. The first part is dithering. The second part is error diffusion. Those two parts are using the grey-scale image. The third part is color halftoning with error diffusion. In this part, we are going to use colorful image.

a. Dithering

1. Fixed thresholding

In this part of the project, I will choose one fixed value T as the threshold. The formula below is separate the pixel value into two different range. I tried T=128. Figure 12 shows the image with fixed threshold T=128.

$$output(i,j) = \begin{cases} 255 & if\ 256 > input(i,j) > T \\ 0 & if\ \ T > input(i,j) > 0 \end{cases} \quad (7)$$

2. Random thresholding

In this part of the project, I will change a fixed value T to a random value T as our threshold. The formula blow is separate the pixel value into two different parts. For each pixel, there is a random number in range 0 to 255. The random number is from the uniformly distributed random variables. I use function rand to generate the uniformly distributed random variable. Figure 13 shows the image with random threshold.

$$output(i,j) = \begin{cases} 255 & if\ 256 > input(i,j) > rand(i,j) \\ 0 & if\ \ rand(i,j) > input(i,j) > 0 \end{cases} \quad (8)$$

3. Dithering Matrix

In this part of the project, the dithering parameters matrix is given by the instructor. The initial index matrix is I2(I,j)=[1 2 ; 3 0]. Then I will use this I2 to generate the T2 threshold matrix. the formula below to calculate the T threshold matrix by using the I2. Then I use formula 7 to determine the output pixel values.

$$T(x,y) = \frac{I_N(x,y) + 0.5}{N\text{^}2} * 255 \quad (9)$$

After I generate I2, I will use I2 to generate I4 by using the formula below. And use the same formula 9 to get T4. After I repeat server time, I will use I2, I8, I32 as my threshold matrices and apply those matrices to the light house image. Figure 14a shows the image with I2 threshold matrix. Figure 14b shows the detail of the I2 threshold matrix. Figure 15a shows the image with I8 threshold matrix. Figure 15b shows the detail of the I8 threshold matrix. Figure 16a shows the image with I32 threshold matrix. Figure 16b shows the detail of the I32 threshold matrix

$$I_{2N}(x,y) = \begin{bmatrix} 4 * In(i,j) + 1 & 4 * In(i,j) + 2 \\ 4 * In(i,j) + 3 & 4 * \mathrm{In(i,j)} \end{bmatrix} \quad (10)$$

b. Error Diffusion

The second method called error diffusion; we will add the diffuse error to the neighbor pixel. There are three different error diffusion. The first one is Floyd-Steinberg's error diffusion. The matrix is 1/16[0 0 0; 0 0 7; 3 5 1]. We also do the serpentine scan for our image. Therefore, the row with even number will use matrix 1/16[0 0 0; 7 0 0; 1 5 3]. The row with odd number still uses the first matrix above. I used two different matrices to store the output data and changing matrix data. First step of the experiment is determining the pixel value by using the equation 7. The threshold is 127 in my experiment. Then we are going to calculate the error by subtracting the original pixel value with the new output pixel value. Using this error to multiple with our error diffusion matrix. Finally, adding this processed matrix to the original neighbor pixel value. repeating the process for each pixel. Figure 17 shows the processed image with Floyd-Steinberg's error diffusion matrix.

The second error diffusion proposed by Jarvis, Judice and Ninke. The matrix is 1/48[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]. Repeating the same process with Floyd-Steinberg's error diffusion. Figure 18 shows the processed image with JJN's error diffusion matrix.

The third error diffusion proposed by Stucki. The matrix is 1/42[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]. Repeating the same process with Floyd-Steinberg's error diffusion. Figure 19 shows the processed image with Stucki's error diffusion matrix.

c. Color Halftoning with Error Diffusion

The final part of the halftoning, I will deal with the color halftoning with error diffusion. There are two different error diffusion for our colorful images. For the colorful image, there are 256^3 million colors. We use 8 color channels to represent the colorful image. binary matrix for each channel.

1. Separable Error Diffusion

The first step of this experiment is separate the image into CMY channels. Since we only have RGB channels, we have to convert RGB to CMY. The formula below is convert the RGB to CMY channel.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad or \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \qquad (12)$$

Then, apply the Floyd-Steinberg's error diffusion algorithm to do the quantize to CMY channel. After I processed the image, I use the equation 12 for convert the CMY back to RGB to display the image. Figure 20 shows the image with separable error diffusion.

2. MBVQ-based Error Diffusion

For the MBVQ-based error diffusion method, we separate the RGB color space into small brightness variation quadrants. There are RGBK, WCMY, MYGC, RGMY, RGBM and CMGB. Figure 21 shows the algorithm of determine the quadrants. After I find the quadrants, I will use the open source that proved by the instructors to find the vertex v. Vertex V is the closed pixel with our current pixel. Then we will use this vertex V to do the error diffusion for each three channels. Figure 22 shows the processed image with MBVQ.

Experimental Results

Figure 12 shows the image with fixed threshold T=128.

Figure 13 shows the image with random threshold.

Figure 14a shows the image with I2 threshold matrix.

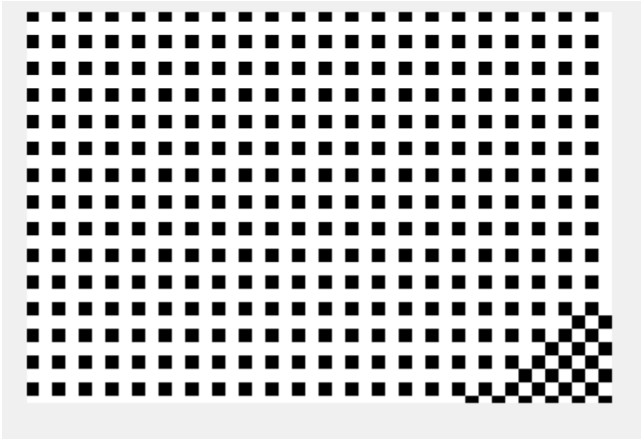


Figure 14b shows the detail of the I2 threshold matrix.

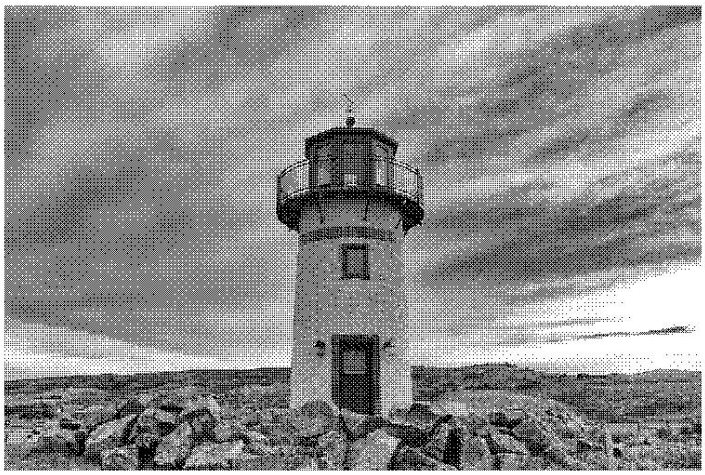


Figure 15a shows the image with I8 threshold matrix.

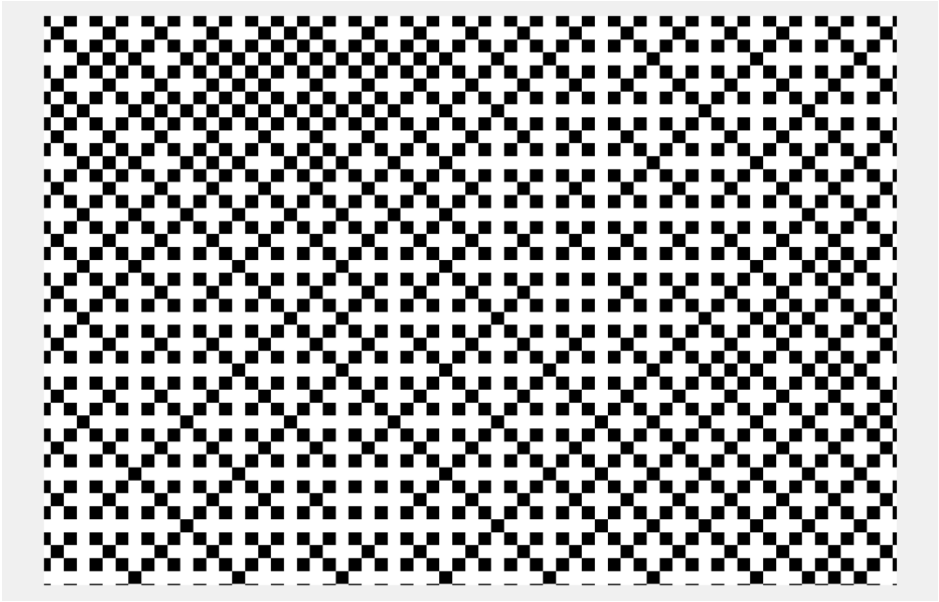


Figure 15b shows the detail of the I8 threshold matrix.

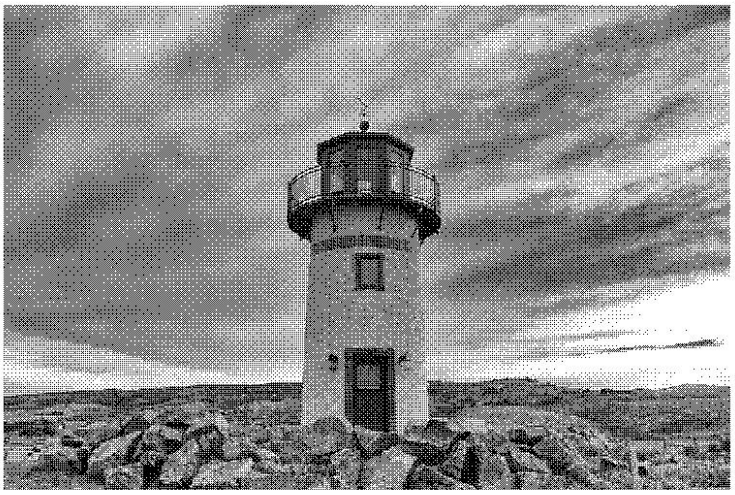


Figure 16a shows the image with I32 threshold matrix.

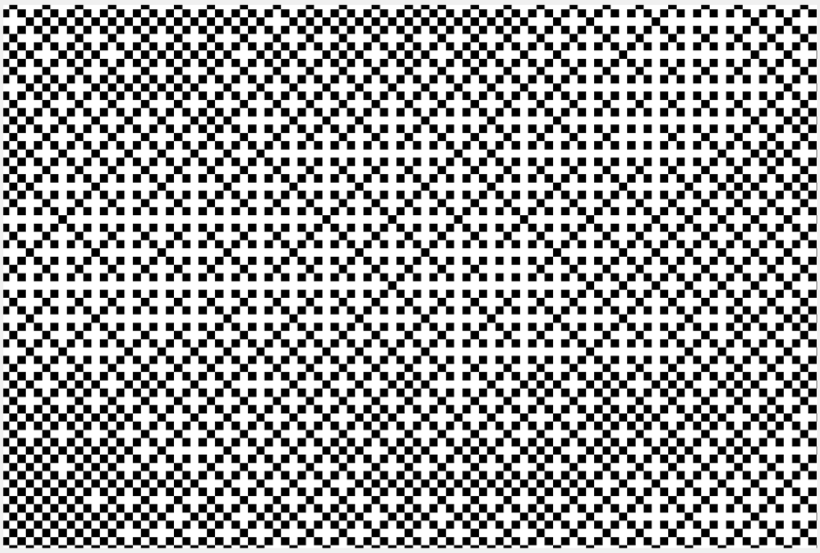


Figure 16b shows the detail of the I32 threshold matrix

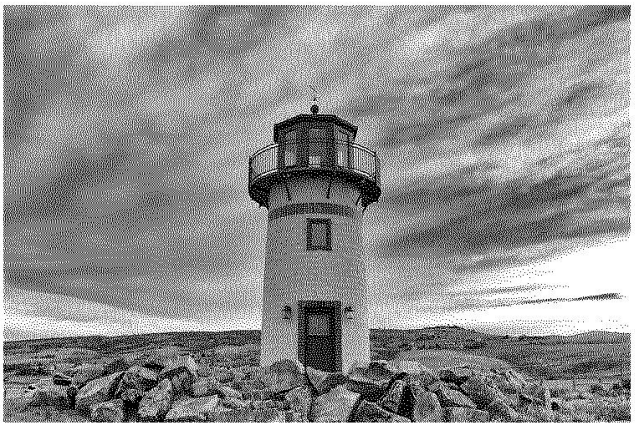


Figure 17 shows the processed image with Floyd-Steinberg's error diffusion matrix.

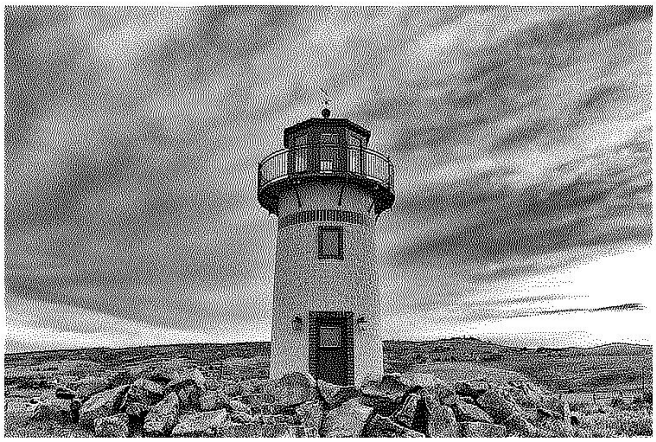


Figure 18 shows the processed image with JJN's error diffusion matrix.

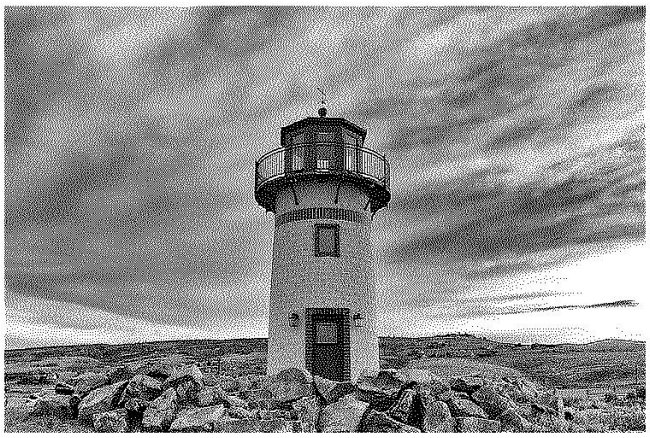


Figure 19 shows the processed image with Stucki's error diffusion matrix.

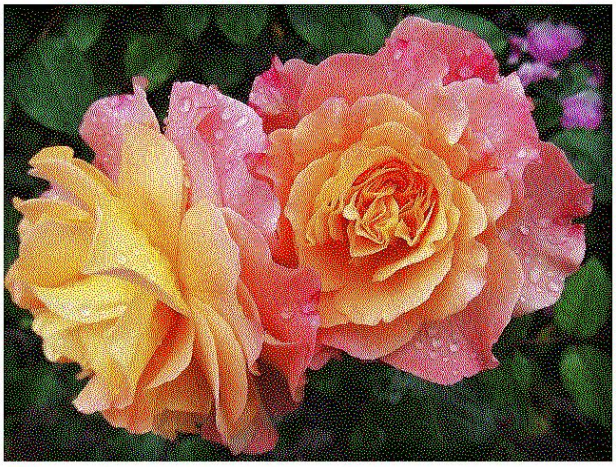


Figure 20 shows the image with separable error diffusion.

```
+3  Lighthouse_dithering.m | error_diffusion_1.m | color_halftoning_
1   function mbvq = return_MBVQ( R, G, B)
2
3 -  if(R+G>255)
4 -      if(G+B)>255
5 -          if(R+G+B) >510
6 -              mbvq ='CMYW';
7 -          else
8 -              mbvq = 'MYGC';
9 -          end
0 -      else
1 -          mbvq = 'RGMY';
2
3 -      end
4
5 -  else
6 -      if ~((G+B)>255)
7 -          if ~((R+G+B)>255)
8
9 -              mbvq = 'KRGB';
0 -          else
1 -              mbvq = 'RGBM';
2
3 -          end
4 -      else
5
6 -          mbvq = 'CMGB';
7 -      end
8
9 -  end
0
1 -  end %function
```

Figure 21 shows the algorithm to determine the quadrant.

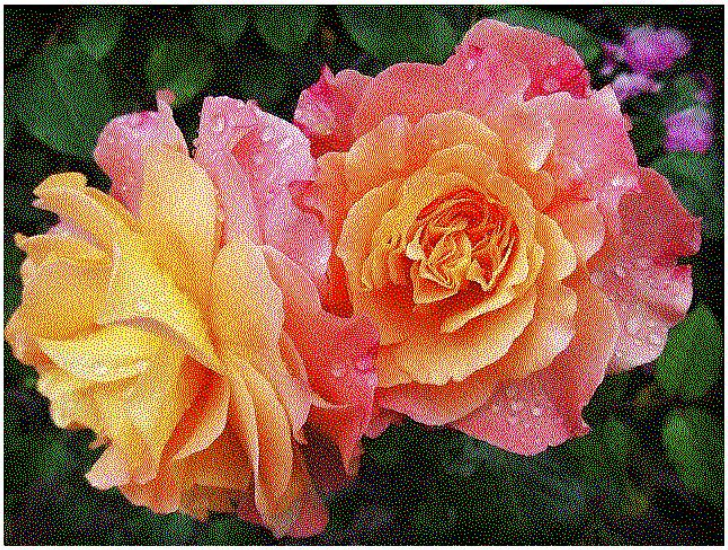


Figure 22 shows the image with MBVQ.

III. Discussion

2a

Compare fixed thresholding, random thresholding and dithering matrix:
For the fixed thresholding, it is obvious, it only contains the white and black area, there are not too much detail for fixed thresholding, but we still can see the majority parts of edge. Compare to the random thresholding, fixed thresholding had a better performance in our problem. There are many random black and white for random thresholding. The image quality become worse. From the visual perspective, I think I32 is the best result among those three threshold matrices. From Figure 16b, I zoomed in the image, I32 give us a better uniform and higher density image. The pattern for I32 are more density. From Figure 14b,15b, the pixels are still proved a better performance compare to fixed thresholding and random thresholding, I32 is the best image for the case. I32 focus on the bigger range of the image. I2 and I8 only filter small range of the image for each time. From the Figure 14b, 15b, 16b, I think the program do correct way.

2b
Compare three different variations:
Figure 17,18,19 shows the image with different variations. There are not big different for each variation. From the Figure 17, there are some white line across the image, it maybe causes by the image with small error diffusion matrix. Figure 18 and 19 didn't have this issue. I think JJN and Stucki have similar result. Stucki is a little bit dark than JJN. But there are not big different.
Compare those three variations with dithering matrix method in part(a):
From my visual perspective, I think error diffusion methods have better image performance. From the Figure 16a, there are obvious artifact texture. But those three variations have a smooth image. I think the reason is because the error diffusion method will affect the neighbor pixel when I do the process. It helps the image with a better performance.
I prefer error diffusion, because error diffusion give us a better solution for our problem, also error diffusion will make the image smoother, since each pixel value will affect its neighbor pixels.
To give a better result:
As the instructor method in the class, using evenly distribute the numbers. The evenly distribute give us a smoother image, because it uses global threshold with uniform distribution. It will make a better result. Or I think if we use Hilbert curve instead of serpentine parsing for our question, it may get a better result.
2c-1
Figure 20 shows the result. There are many random colorful dots in the image. The image is not smooth as the original image, but we still can see the detail of image very clearly. I think the main shortcoming of this approach is the image will become more blurred, because we have a fix CMY value for the result. As I will discussion next part, MBVQ will overcome the shortcoming of this issue. There are many random colorful dots around each point. I think the reason of those random dot is because we reduce the 256^3 colors to the less colors.

2c-2
The key ideas of MBVQ-based error diffusion method are finding the short distance to each concern of the MBVQ. It will prove more different color values for each point. For the separable error diffusion method, we only have a fixed CMY values. But MBVQ have 8 different values for the image.

Compare MBVQ with separable error diffusion method:
There are not big different between two images from the visual perspective. I think it's because the resolution for the image will affect our result. But MBVQ is slightly better than separable error diffusion.