Samwoo Seong
7953-6137-66
samwoose@usc.edu

EE569 Homework #2
February 14, 2020

1.

1.1 Abstract and Motivation

(a), (b), (c), (d)

In images, there are some certain information human cares about. One of them is what is an object or not. Historically, scientists and engineers have tried to obtain this information in several ways because it can be employed in many different applications such as autonomous vehicles that use computer vision to detect objects. This task can be called object detection. To accomplish the task, detecting edges plays a key role. During past 40 years, there are many ways developed to detect edges in images. Here, we will implement and discuss traditional and modern edge detection algorithms.

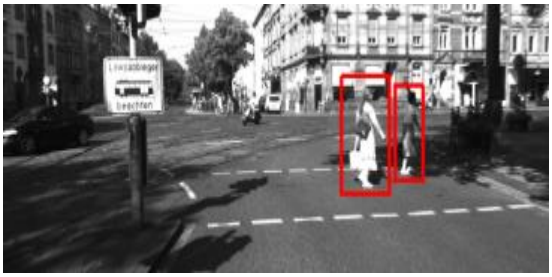


Fig 1. Computer Vision object detection example. Red box captures pedestrians [1].

1.2 Approach and Procedures

(a)

(a-1) Approach: Sobel Edge detection algorithm is one of the simplest algorithms. Underlying principle is that around boundary between background and target object, it tends to have a big change in terms of pixel value, therefore, this algorithm is trying to capture the locations where have relatively huge gradient values in different direction. This idea can be accomplished by 2 gradient masks shown below. First one is to calculate gradient in horizontal direction and second one is to calculate gradient in vertical direction.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Fig 2. Sobel filters. Filter for horizontal direction(left), Filter for vertical direction(right) [2]

(a-2) Procedure: the whole procedure can be summarized with a flowchart below.

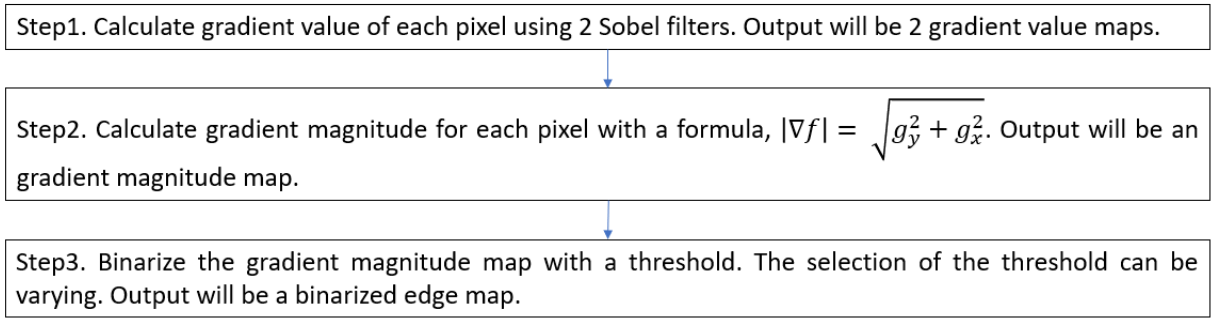


Fig 3. Flowchart of Sobel Edge detection procedure.

(b) [Online Source Code]

(b-1) Approach:

Canny edge detection method can be considered as upgrade version of Sobel edge detection algorithm [3]. In this method, it has "extra steps" compared to Sobel edge detection method. The first step is called Non-Maximum Suppression, and the second step is called Hysteresis Thresholding as known as Double Thresholding. Role of Non-Maximum Suppression is to make thick edges to thin edges. This can be done by eliminating non maximum gradient value in certain angles to zero and remaining maximum gradient. The second step is for handling a certain type of noise such as impulse noise. Briefly speaking, the probability of edge that is greater than high threshold is considered as strong edge. On the other hand, probability of edge that is lower than low threshold is called weak edge. Strong edges remain at final processing, but weak edges become zero. For those probability lying on area between high and low thresholds, they become edges in case they are connected to strong edges. Otherwise, they are eliminated.

(b-2) Procedure:

Canny edge detection technique can be shown as a flowchart.

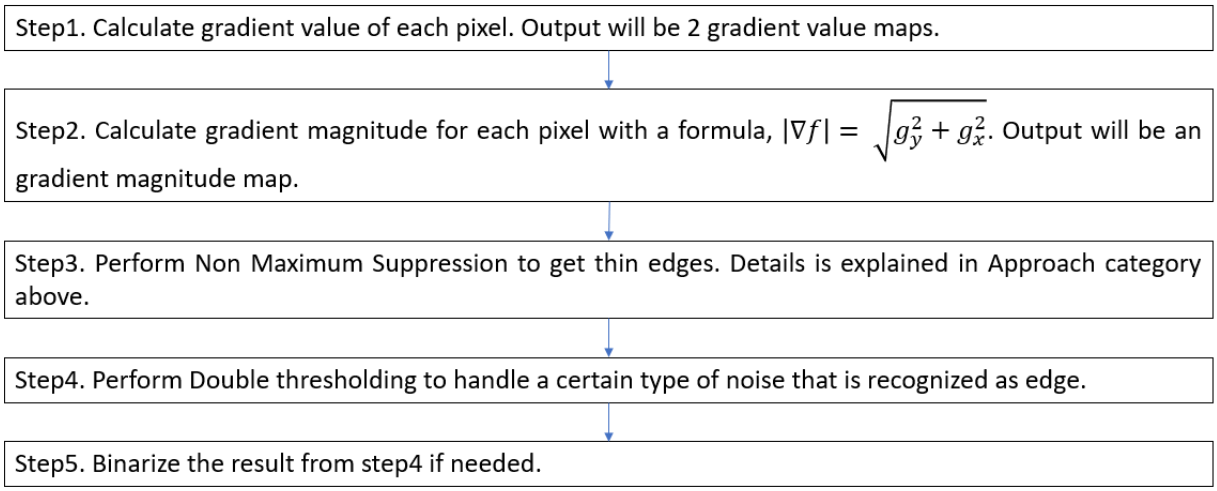


Fig 4. Procedure of Canny edge detection method [3].

(c) [Online Source Code]
(c-1) Approach:
We have been using non-data driven methods such as Sobel and Canny Edge detections. However, edge detection can be achieved by data driven method, so called, Structured Edge Detection. This method consists of patch clustering, extracting features, training machine learning model as known as Random Forest, and testing. Specifically, testing process will provide us an edge map of an input image. "Random Forest Algorithm is one way to overcome cons of CART method that $var(x)$ regarding error bound can be high where x is training samples.
The reason why Random Forest Method can reduce value of $var(x)$ is that drawn d features in the method are d<D (number of all features) or d<<D. Therefore, amount of correlation between features gets smaller. In consequence, value of $var(x)$ becomes lower. After growing all trees, we can decide decision boundary by majority vote meaning a data point in certain region will be assigned to the class that is mostly assigned by all trees" [5].

(c-2) Procedure:
In Structured Edge Detection, we can summarize whole procedure of the method and Random Forest with two flowcharts below.

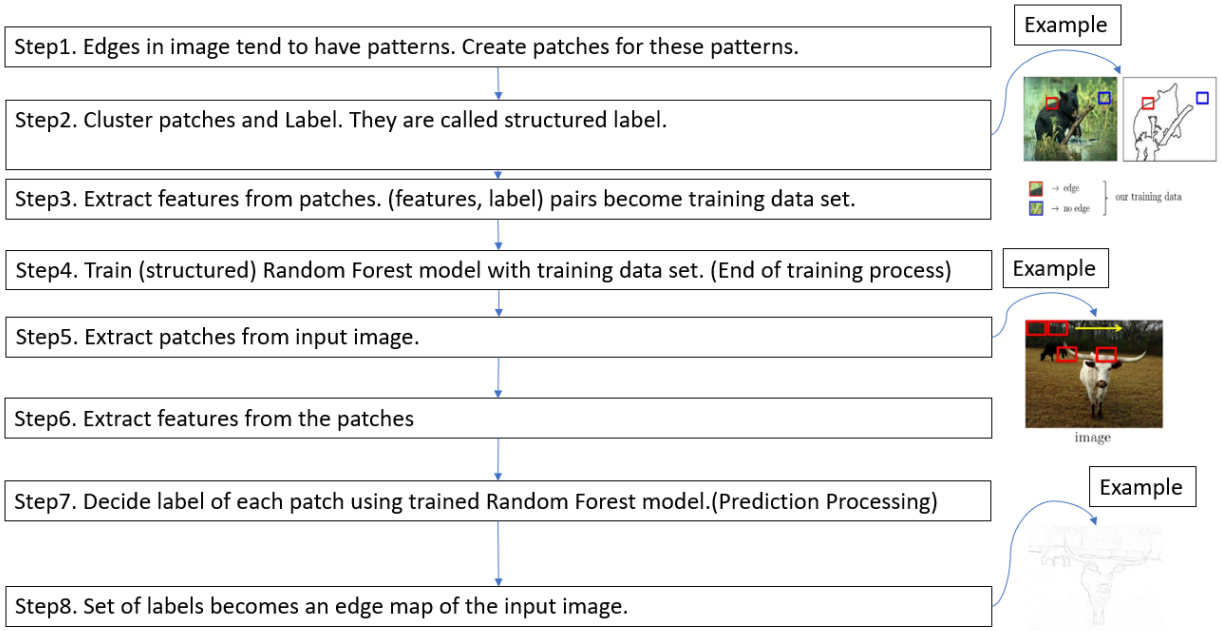


Fig 5. Flowchart of Structured Edge Detection procedure [2], [4]

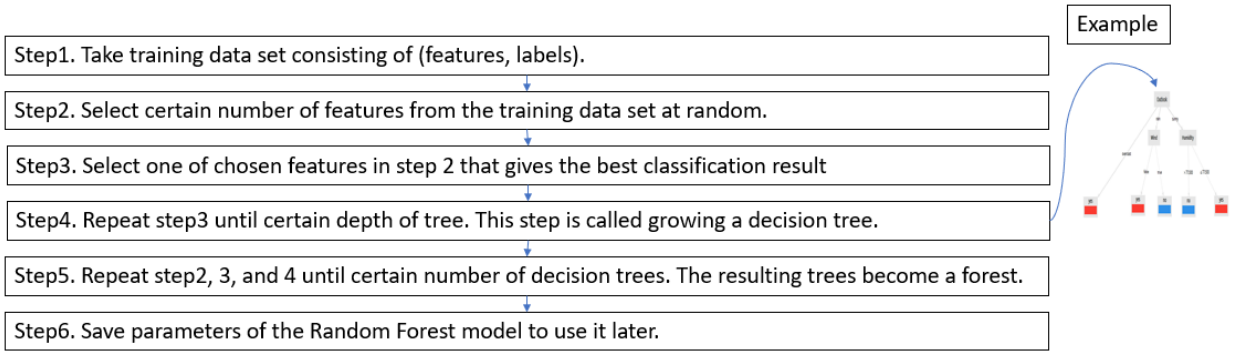


Fig 6. Flowchart of Random Forest procedure [2]

(d) [Online Source Code]

(d-1) Approach:

We can evaluate performance of each edge detection method with several metrics. However, in this problem, we will use F score metric. F score includes recall and precision metrics. Purpose of recall is to measure effect of false negative, while precision is to measure effect of false positive. F score is invented to find a balance between these two metrics.

For the ground truths, individuals could have different observations. Therefore, we average 5 ground truths to calculate F score.

One thing we need to bear in mind is that provided MATLAB code “edgesEvalImg” is using distance measuring method meaning it consider a pixel as true if that is within certain distance from pixel in ground truth.

(d-2) Procedure: performance evaluation procedure can be shown as follows.

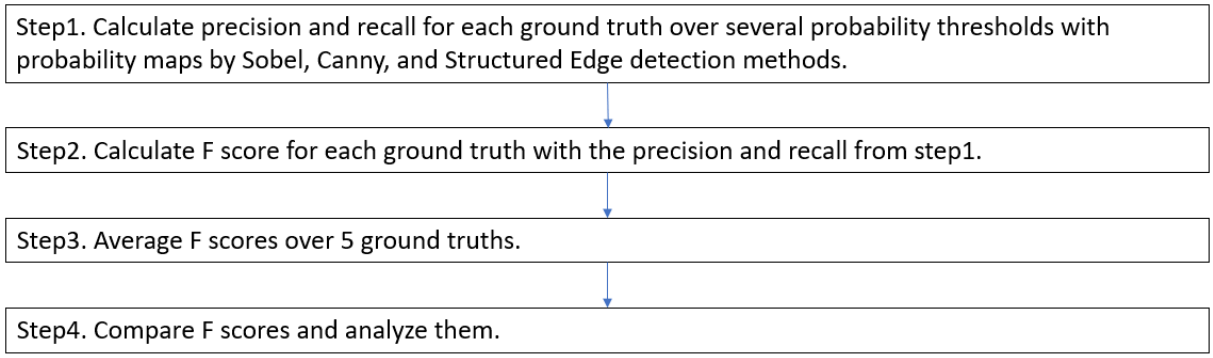


Fig 7. Flowchart of performance evaluation procedure.

1.3 Experimental Results

(a)

(a-1)

Dogs.raw

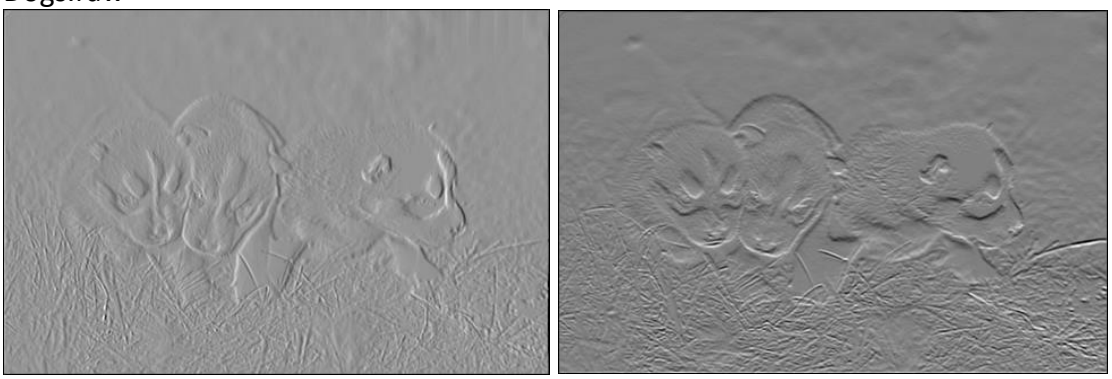


Fig 8. Normalized x-gradient edge map(left), normalized y-gradient edge map(right)

Gallery.raw

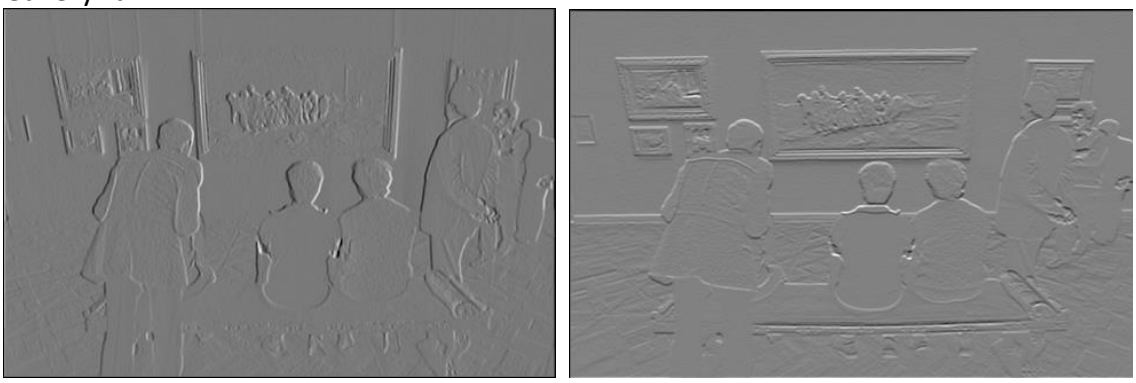


Fig 9. Normalized x-gradient edge map(left), normalized y-gradient edge map(right)

(a-2)

Dogs.raw

Fig 10. Normalized gradient magnitude map

Gallery.raw

Fig 11. Normalized gradient magnitude map

(a-3)

Dogs.raw

Fig 12. Tuned edge map with 90% of maximum gradient magnitude. (inversed)

Gallery.raw

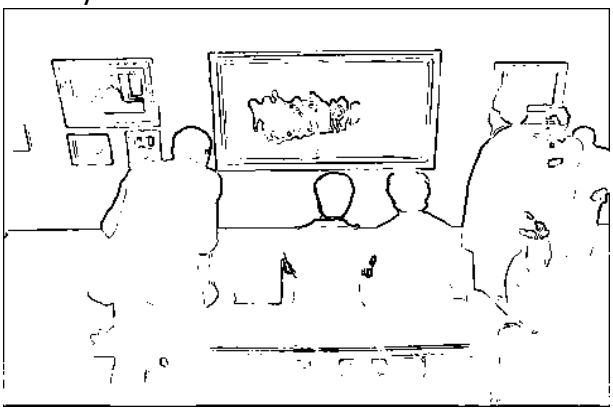


Fig 13. Tuned edge map with 90% of maximum gradient magnitude. (inversed)

(b)

Dogs.raw

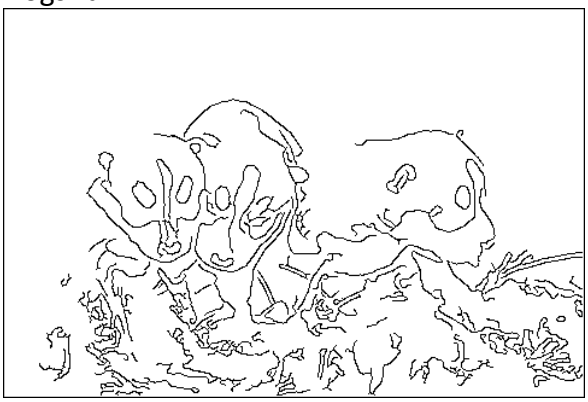


Fig 14-1. Edge map by Canny method with low = 0.1, high = 0.35 (inversed)

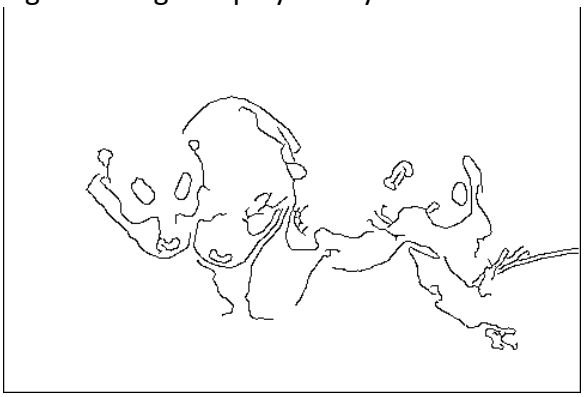


Fig 14-2. Edge map by Canny method with low = 0.1, high = 0.6 (inversed)

Gallery.raw

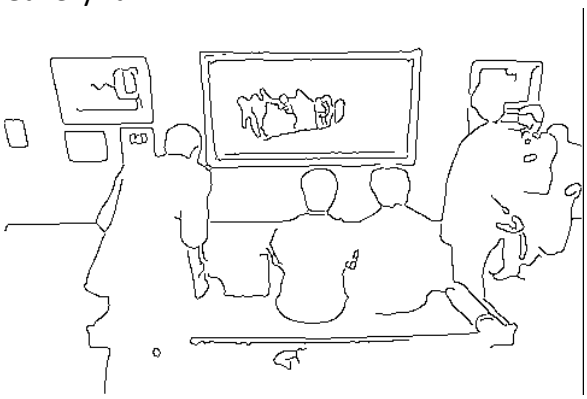


Fig 15-1. Edge map by Canny method with low = 0.1, high = 0.35 (inversed)

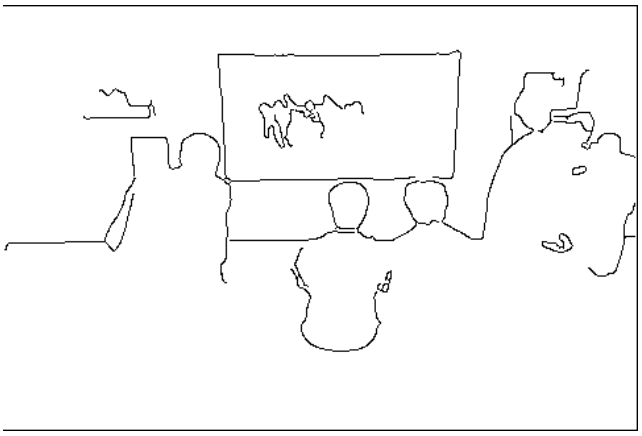


Fig 15-1. Edge map by Canny method with low = 0.1, high = 0.6 (inversed)

(c)

Dogs.raw

Fig 16-1. Edge map by Structured Edge detection method with pretrained RF model(inversed)

Gallery.raw

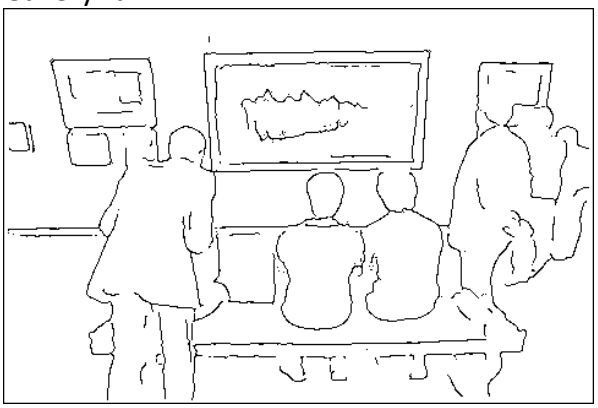


Fig 16-2. Edge map by Structured Edge detection method with pretrained RF model(inversed)

(d)

	Sobel	Canny	Structured Edge
Value of chosen threshold	0.9	0.8	0.9
Average Precision over 5 ground truths	0.2379	0.1015	0.8302
Average Recall over 5 ground truths	0.4449	0.7910	0.1456
Average F score over 5 ground truths	0.2379	0.1799	0.2477

Table 1-1. Chosen threshold and mean F score for each method (Dogs.raw).

	Sobel	Canny	Structured Edge
Value of chosen threshold	0.7	0.3	0.6
Average Precision over 5 ground truths	0.4248	0.2157	0.8222
Average Recall over 5 ground truths	0.7496	0.7312	0.4120
Average F score over 5 ground truths	0.5423	0.3332	0.5489

Table 1-2. Chosen threshold and mean F score for each method (Gallery.raw).

I used 9 probability thresholds (e.g. 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9) because of computational cost.

1.4 Discussion

(b)

It is each to observe that the resulting edge maps are different depending on the chosen two threshold values. This is because high threshold tends to cause smaller number of pixels considered as edges in the input image. Furthermore, we can see a little bit of enhancement in the results compared to the results from Sobel edge detection since double thresholds process can remove potential edges caused by texture (e.g., grass in Dogs.raw) which is not a target object such as dogs and people in the images.

(c)

I set multiscale to 0 not to run multiscale edge detector. For sharpen and nThreads, I used default parameters as given. However, nTreesEval(i.e., number of trees to evaluate per location) is set to 100 because it often lead a trained model to better prediction in Random Forest algorithm. Also, I don't want to set this parameter too high because of computational cost that might be expensive. For the comparison purpose with the edge maps from Canny method, I set nms to 1 meaning perform Non-Maximum Suppression during edge detection process. The value for each parameter is summarized in Table 2.

	multiscale	sharpen	nTreesEval	nThreads	Nms
Chosen value	0	2	100	4	1

Table 2. Chosen parameters for Structured Edge Detection method

Speaking of visual results, the results from Structure Edge Detection method capture edges of interesting objects such as dogs and people in each input image. On the other hand, to extract edges of dogs from Dogs.raw by Canny method, edge of grass is also captured because it is considered edges if algorithm uses a gradient based method. Therefore, if there are a lot of texture in background of an image, Structured Edge Detection method could perform better in terms of edge detection task.

(d)

(d-1)

	Pros	Cons
Sobel Edge Detection	-Can be applied for any single image -Easy to implement -Fast performance	-Image dependent meaning it might capture texture of background such as grass that is not a target object in Dogs.raw -Need to choose threshold carefully.
Canny Edge Detection	-Can be applied for any single image -Can handle to eliminate some amount of weak edges caused by impulse noise or texture.	-Still it might consider textures of non-target object as edges. -Need to choose two thresholds carefully.
Structured Edge Detection	-Can capture edges of target object relatively well -Fast if model is pretrained	-Need a lot of training data to perform well -Might need ground truth by human which is expensive. -If model is not trained in advance, it can be computationally very expensive.

Table 3. Pros and cons for each edge detection method.

(d-2)

I think it is easier to get high F score with Gallery.raw because it has less texture compare to Dogs.raw. Texture such as grass can be captured as edges and this can increase value of number of False Positive since ground truth generated by human tends to only have interesting object's edges such as puppies or audiences. As a result, relatively high number of False Positive will decrease value of Recall and make difference between Recall and Precision meaning F score will become small.

(d-3)

If Precision increases meaning number of true positive is big and number of false positive is small. It has a chain effect that Recall gets smaller. Therefore, for instance, Let P = 1 and R = 0.1. Since

F score has PxR term in its numerator, the total value of F gets small. Hence, it is impossible to get high F score when value of P is higher than R, or R is higher than P.

The given proof question can be proved as follows.
Let's define variables
F: F score, P: Precision, R: Recall, and C: constant value
Assumption: $\mathrm{R} + \mathrm{P} = \mathrm{C}$
F can be rewriten $\mathrm{F} = \frac{2\mathrm{P}(\mathrm{C}-\mathrm{P})}{C} = \frac{2}{C}(PC - P^2)$.
Maximum value of F will exist at $\nabla\mathrm{F}(\mathrm{P}) = 0$
Let's solve the Equation.
$\nabla\mathrm{F}(\mathrm{P}) = C - 2P = 0$ -> C = 2P ->Plug in this to the assumption.
R + P = 2P -> P = R.
Therefore, if R+P is constant, then F has maximum value when R = P.

2.
2.1 Abstract and Motivation
(a), (b), (c)
In modern society, usage of printers has increased. Hence, printing technology plays a significant role nowadays. A way to represent images on a paper is different from how computer or TV screen shows images to us. In printing, there are only black dot and non-dot in gray scale images and one of black, magenta, yellow, and cyan dots and non-dot in color images. With these limited options, we need to create illusion to human eyes so that we are still able to percept printed images. To create such illusion, we will implement and discuss several digital half-toning methods such as dithering and error diffusion in gray scale and color image both.

Fig 17-0. Original image on computer screen(left), Possible printed image on paper(right) [7]

2.2 Approach and Procedures
(a)
(a-1)
(a-1-1) Approach:

The easiest, but not elegant way to half tone an image is to threshold an image by a fixed threshold. Typically, fixed threshold, T, is 127. If the value of pixel is above T, then it becomes 255. Otherwise, it becomes 0.

(a-1-2) Procedure:
Procedure of fixed thresholding can be shown with a flowchart below.

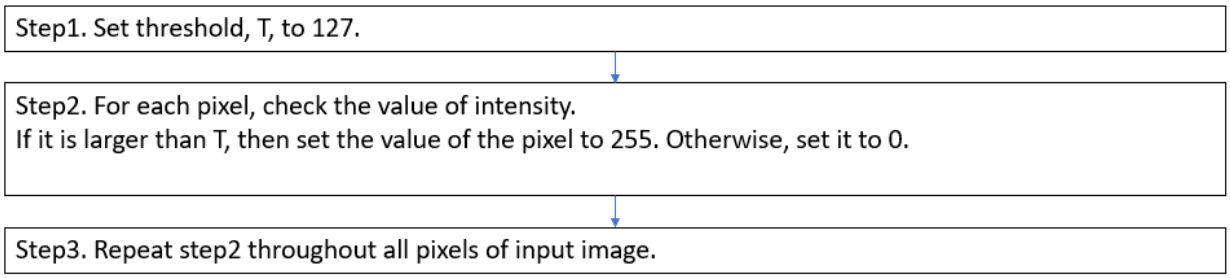


Fig 17. Flowchart of fixed thresholding procedure

(a-2)
(a-2-1) Approach:
Another way to perform dithering is using a random threshold at each pixel. A resulting image can be varying depending on the way random threshold is generated. It has the sample underlying principle as the fixed thresholding except its threshold is generated at random for each pixel.

(a-2-2) Procedure:
Procedure for this method can be illustrated with a flowchart.

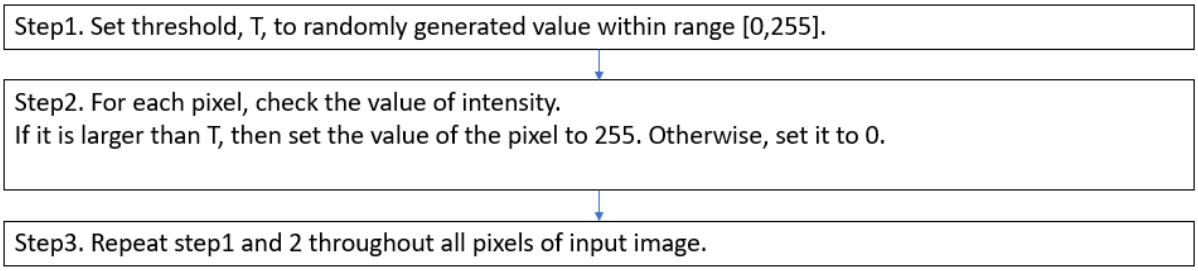


Fig 18. Flowchart of random thresholding procedure

(a-3)
(a-3-1) Approach:
The other way to perform digital half toning in dithering is using index mask and creating threshold mask based on the index mask. The index and threshold mask can be generated recursively in given pattern below. Each value of pixel (x,y) become 255 if it is larger than value T(x,y). Otherwise, it becomes 0.

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

Fig 19. Initial index mask [6]

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

Fig 20. Pattern to expend size of index mask [6]

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

Fig 21. Formula to create a threshold mask based on the index mask [6]

(a-3-2) Procedure:

Procedure for dithering matrix method can be summarized with a flowchart.

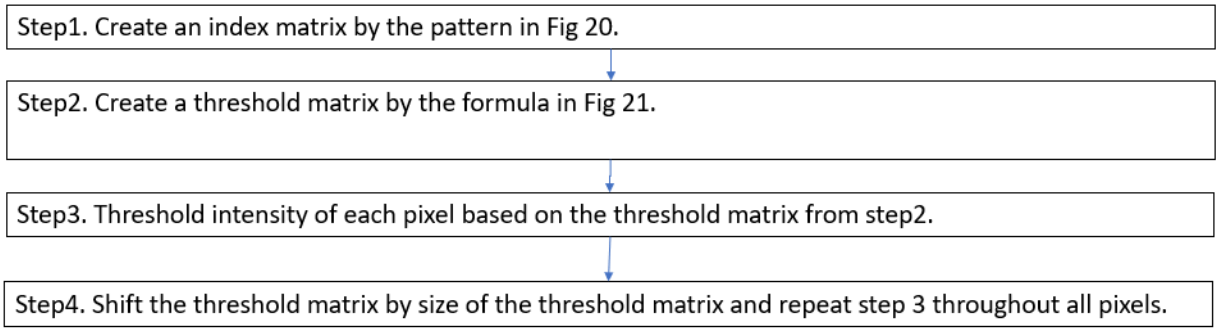


Fig 22. Flowchart of dithering matrix procedure.

(b)

(b-1) Approach:

In digital half toning, error between original intensity of a pixel and binarized intensity of the pixel occurs inevitably. This method is based on the idea that illusion can be created better if the error is distributed to neighbor pixels and binarize their intensity. In this error diffusion method, diffuse error to only “future pixels”. This diffusion is illustrated in figure 23. Also, to prevent error from accumulating at right corner, we can use serpentine scanning along side different error diffusion matrixes such as Floyd-Steinberg’s, JJN’s, and Stucki’s matrix.

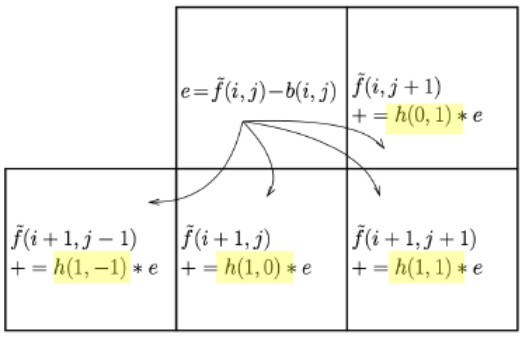
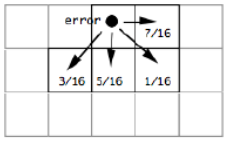
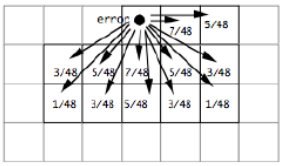


Fig 23. Illustration that explains how to diffuse error [7]

- **Floyd-Steinberg** $\frac{1}{16}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 7\\ 3 & 5 & 1\end{bmatrix}$



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



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

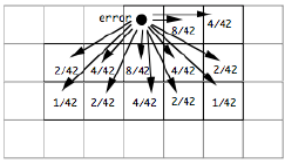


Fig 24. Illustration that explains how to diffuse error for each error diffusion matrix [2]

- **Serpentine parsing**

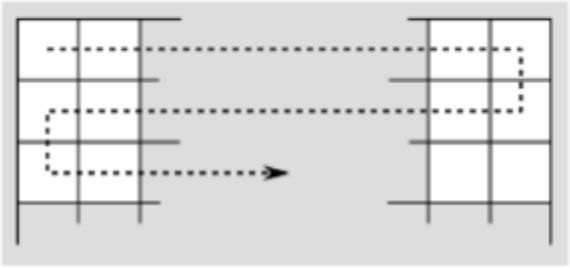


Fig 25. Serpentine parsing illustration [2]

(b-2) Procedure:
Procedure for error diffusion can be illustrated with a flowchart below.

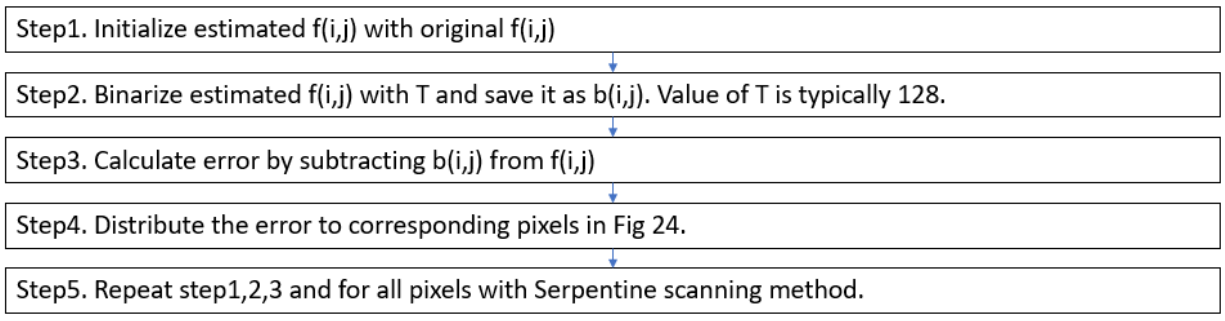


Fig 26. Flowchart of error diffusion method with Serpentine parsing [2]

(c)
(c-1-1) Approach:

Idea of separable error diffusion method is to apply error diffusion with Floyd Stenberg's matrix and Serpentine parsing to each separate color space C, M, Y which can be found by subtracting R, G, B from 255 respectively. Then, combine those three colors spaces in CMY order. For display purpose, we can convert back to RGB space in the same manner.

(c-1-2) Procedure:

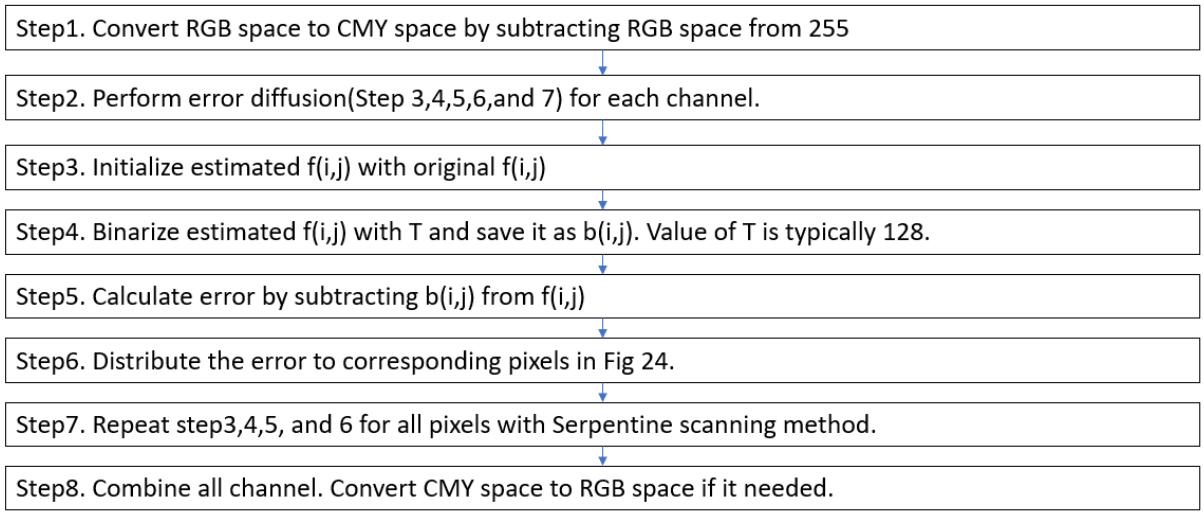


Fig 27. Flowchart of separable error diffusion procedure

(c-2-1) Approach[Provided getNearestVertex.m partially used]:
While separable error diffusion seems fine, there is a shortcoming that it doesn't take into account of correlation of each color space. To overcome the shortcoming, we can come up with idea that we will use whole RGB value for binarization. With a serial of decision trees we can find the closest vertex to original RGB value of a pixel in color space. In this way, we are considering correlation amongst R, G, and B color. Then, we can perform error diffusion step once we find the closest vertex (i.e. binarization of original color)

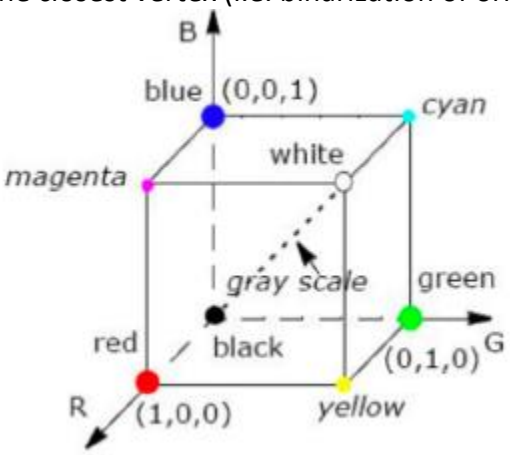


Fig 28. Vertex that represents each color [2]

(c-2-2) Procedure:

Procedure for MBVQ-based color error diffusion is shown below.

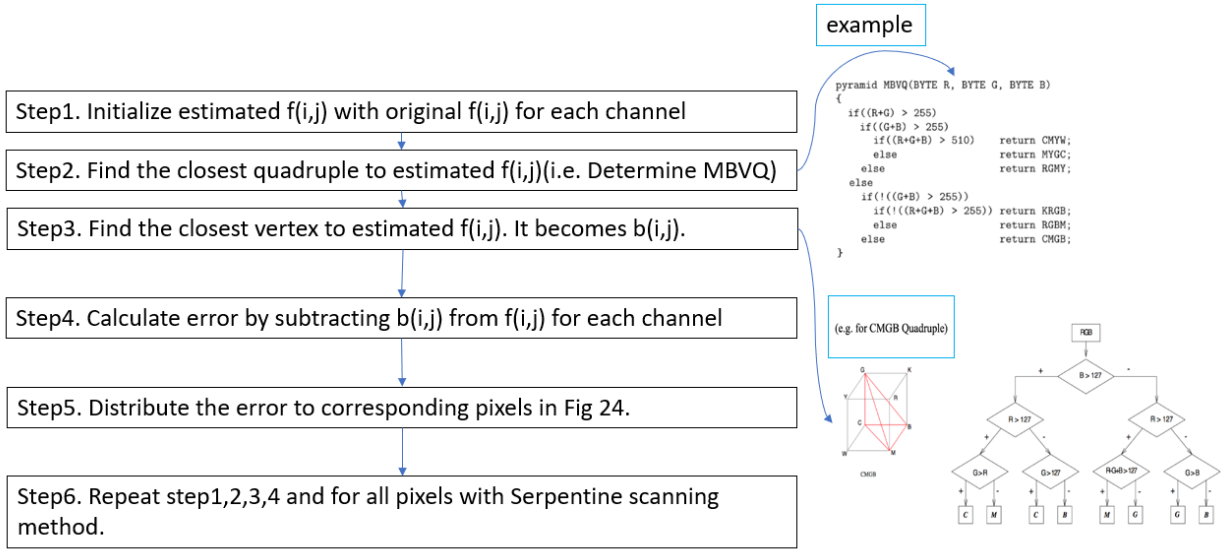


Fig 29. Flowchart for MBVQ-based color error diffusion procedure [2]

2.3 Experimental Results

(a)

(a-1)

Fig 30. Halftoned image by fixed thresholding method

(a-2)

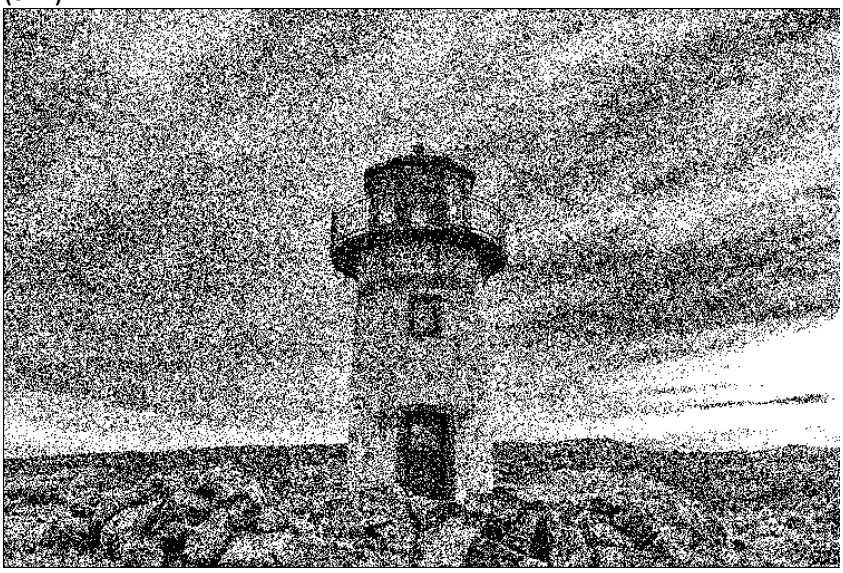


Fig 31. Halftoned image by random thresholding method

(a-3-1)

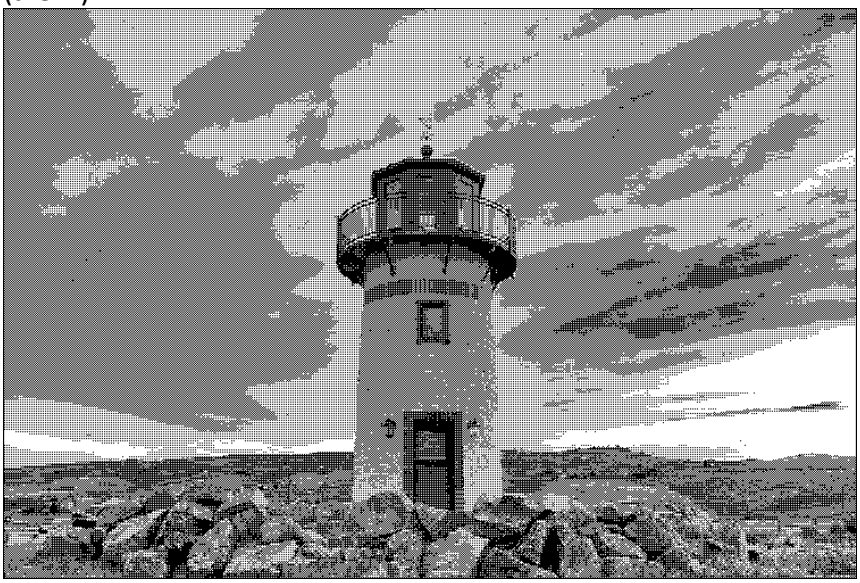


Fig 32. Halftoned image by dithering matrix I_2

(a-3-2)

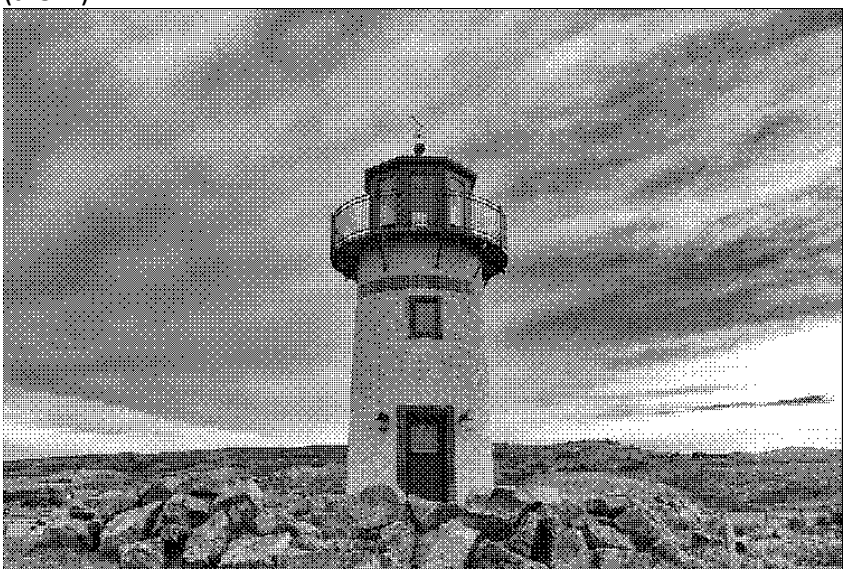


Fig 33. Halftoned image by dithering matrix I_8

(a-3-3)

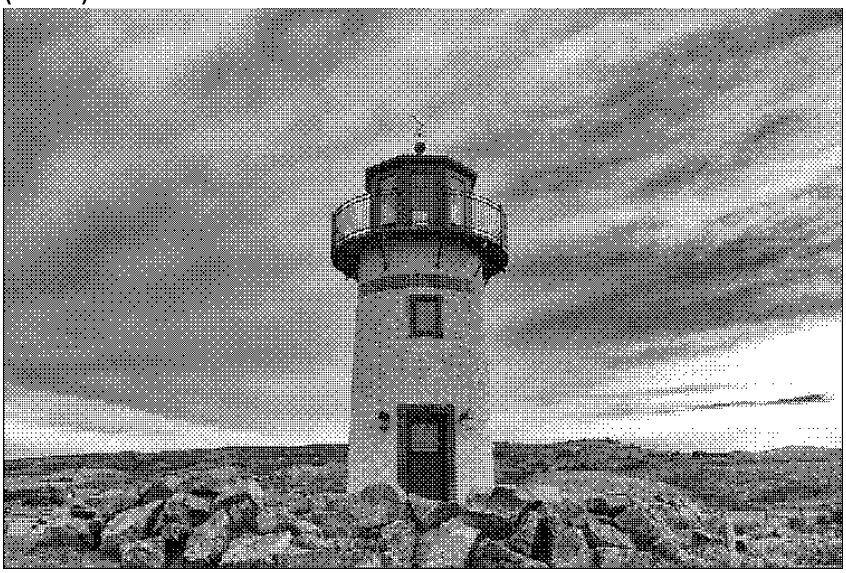


Fig 34. Halftoned image by dithering matrix I_{32}

(b)

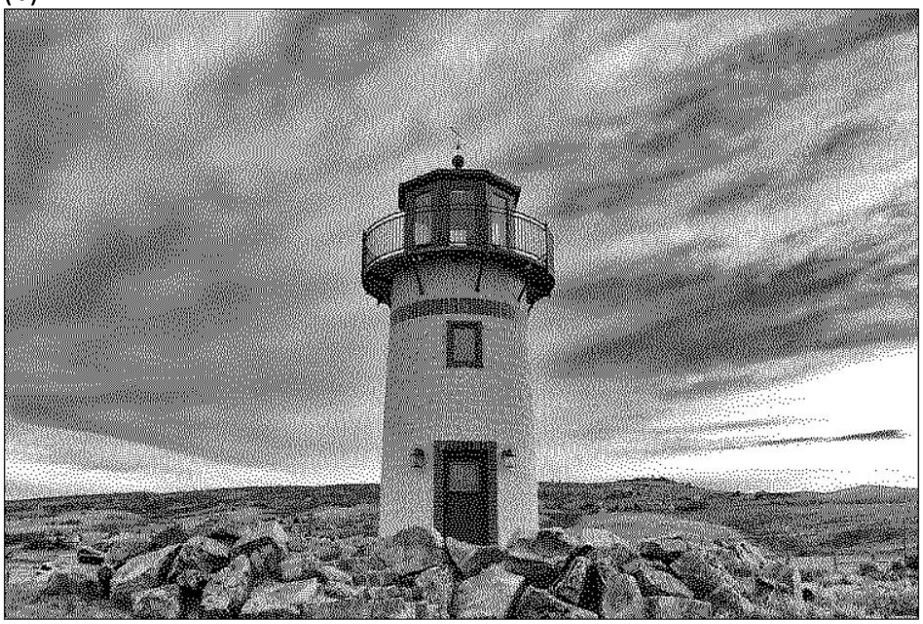


Fig 35-1. Halftoned image by error diffusion method with Floyd Stenberg's mask

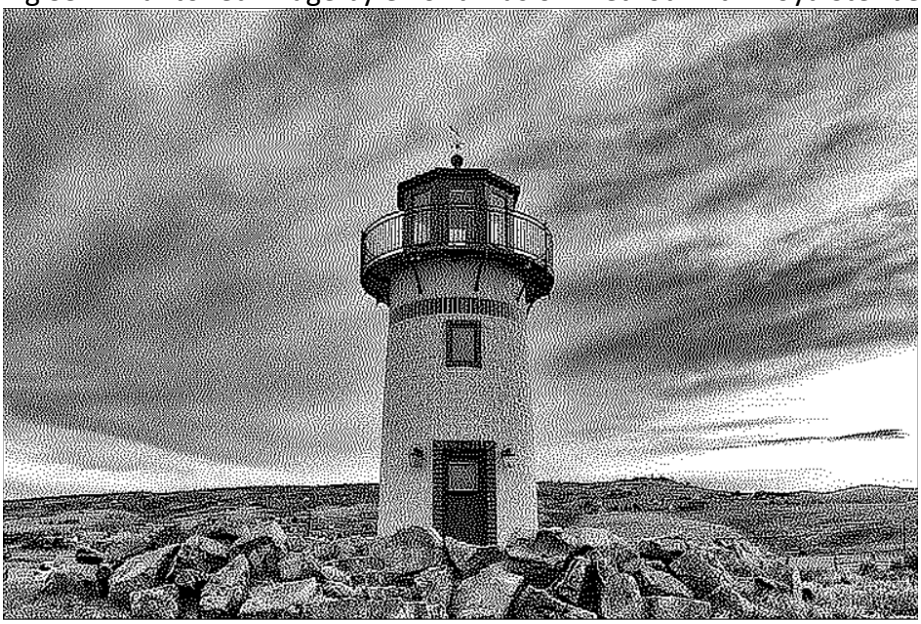


Fig 35-2. Halftoned image by error diffusion method with JJN's mask

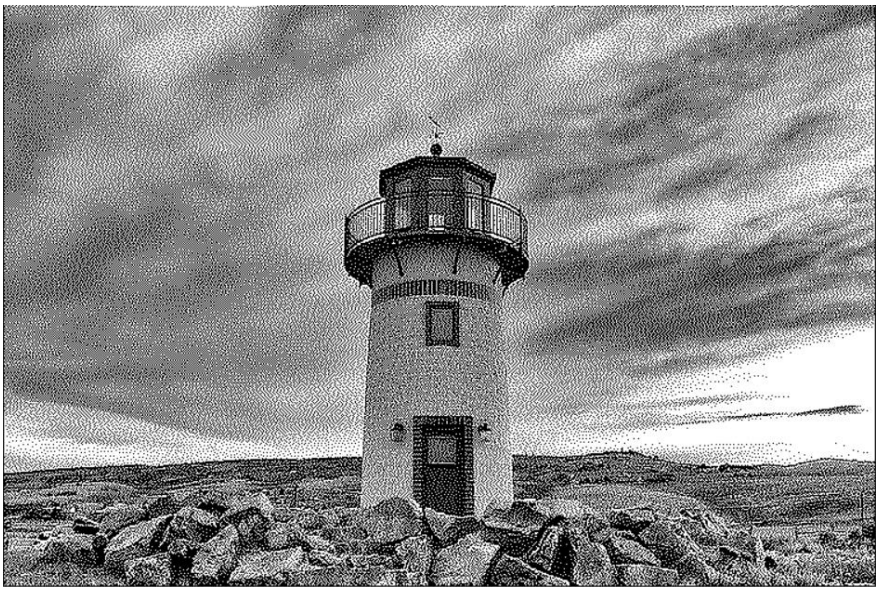


Fig 35-3. Halftoned image by error diffusion method with Stucki's mask

(c)

(c-1)

Fig 36. Halftoned image by separable color error diffusion method

(c-2)

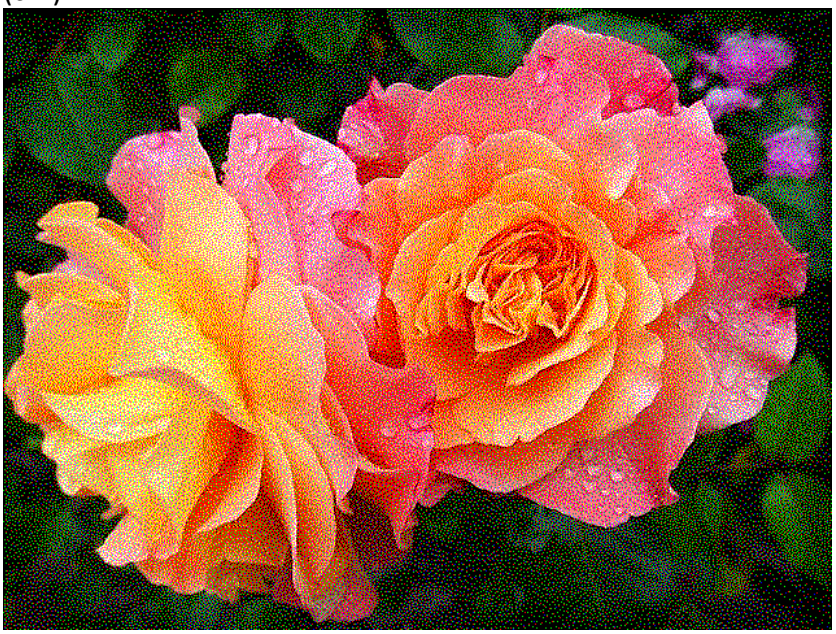


Fig 37. Halftoned image by MBVQ-based color error diffusion method

2.4 Discussion

(a)

For the result by fixed thresholding method, it couldn't make much of illusion to human eyes because of simplicity of the algorithm.

Halftoned image by random thresholding method seems it has random noise. This is because thresholding by random value within a range [0,255] can act like random noise. In this implementation, random value is created uniformly.

As we can observe in Fig 38,39, and 40, the larger index matrix we employ in halftoning, the more 'x' marks appear on the results. This is because of how we expand our index matrix. The index matrix is expanded by setting previous index matrix in crossing pattern. However, despite of x crossing pattern, the larger dithering matrix provides us relatively pleasant illusion of image.

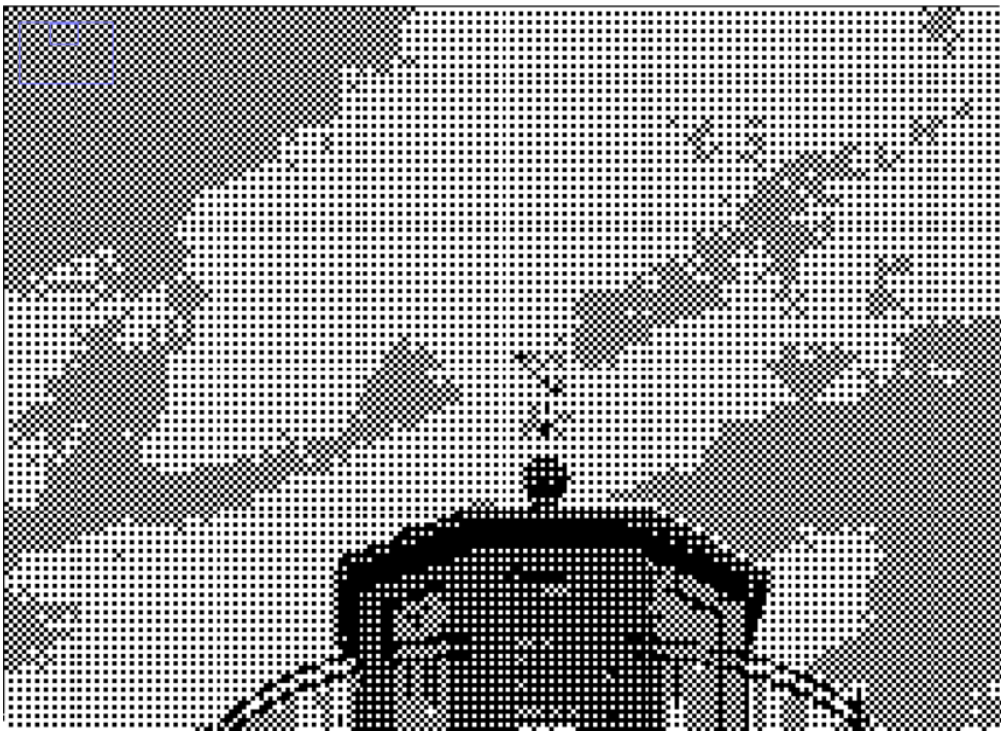


Fig 38. Zoom in halftoned image by dithering matrix I_2(297%)

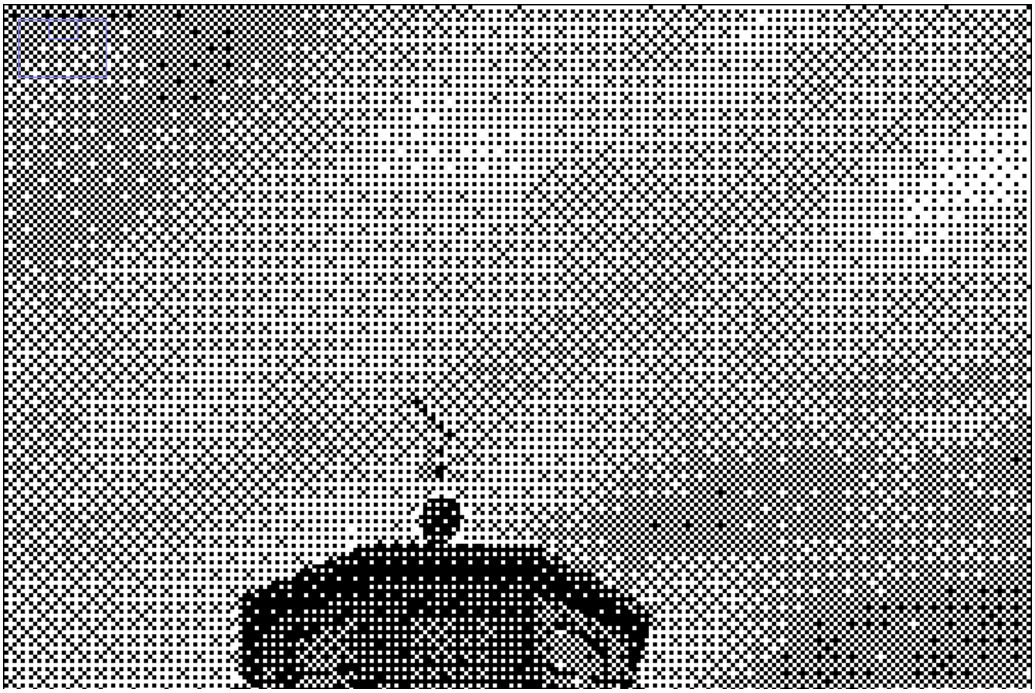


Fig 39. Zoom in halftoned image by dithering matrix I_8(300%)

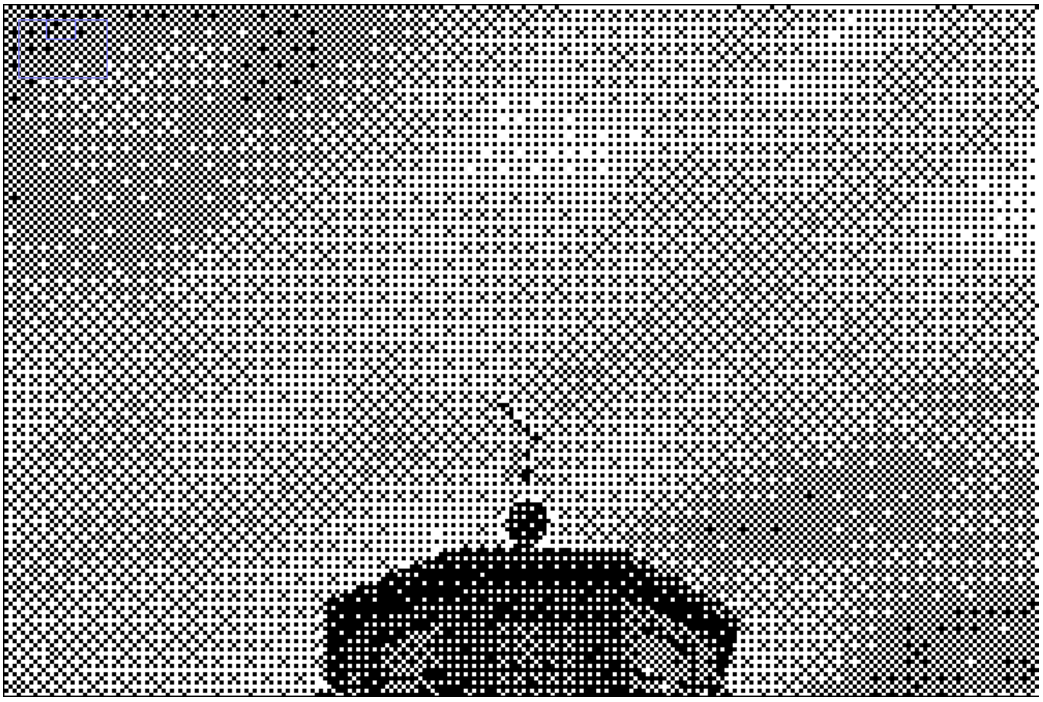


Fig 40. Zoom in halftoned image by dithering matrix I_{32}(300%)

(b)

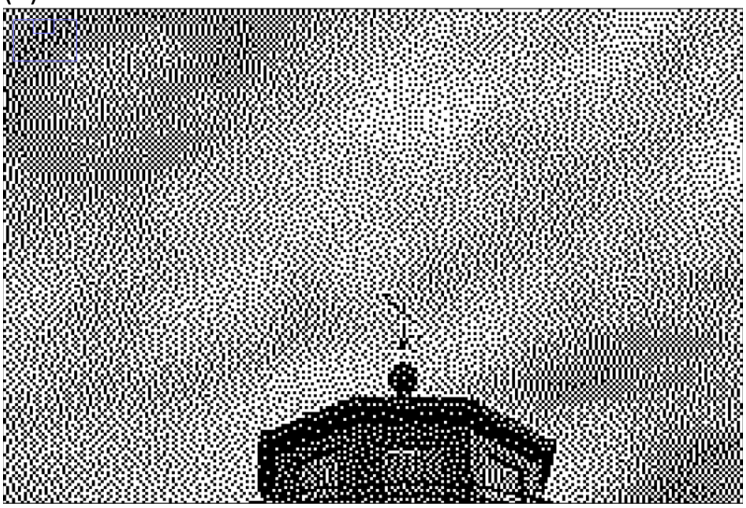


Fig 41-1 Zoom in halftoned image by Floyd mask (300%)

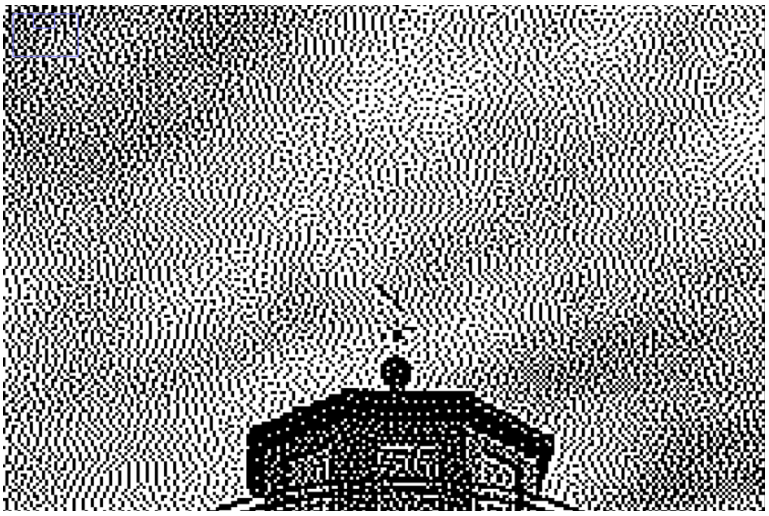


Fig 41-2 Zoom in halftoned image by JJN mask (300%)

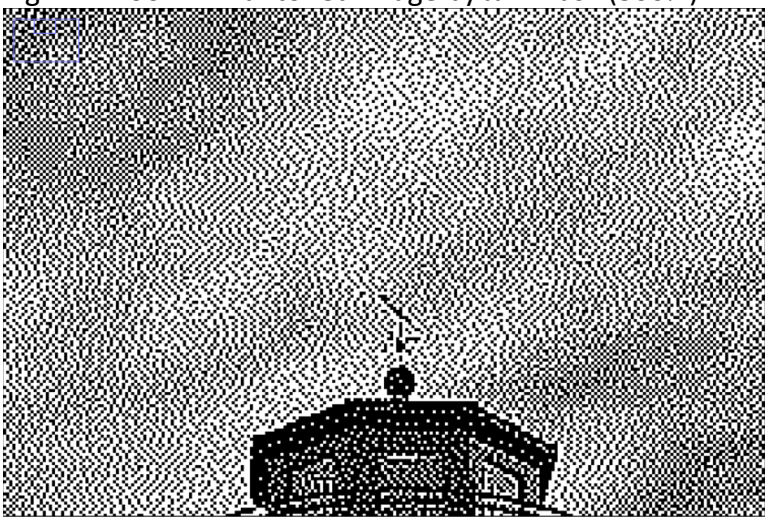


Fig 41-3 Zoom in halftoned image by Stucki mask (300%)

As we can observe above, the artifact as known as x marks disappears with error diffusion method. However, there is some pattern called snake shape.

I would prefer Floyd-Steinberg's error diffusion method to other 2 methods. This is because the halftoned image by Floyd-Steinberg's mask has less severe snake shape artifact compared to the other 2 results.

For my own idea to improve half toned image, I would try to apply Floyd's mask forward and backward for each column. In that way, I can distribute error more uniformly to surrounding pixels.

(c)

(c-1) The main shortcoming of separable error diffusion approach is that it doesn't account for correlations amongst R, G, and B color space when it binarizes pixel values of a color image. It can cause some artifact dots in printed image.

Fig 42. Pure color dots (i.e., artifact of this method) in the color halftoned image (300% zoom in)

(c-2) In MBVQ-based error diffusion approach, it binarizes pixel values of a color image with whole R, G, and B values by determining the closest vertex to the original pixel. In this manner, it causes an effect that it considers correlations amongst all color space. Therefore, it can reduce shortcoming of separable error diffusion method and provide relatively pleasant illusion to human eyes.

References

[1] Faizan Shaikh, “Understanding and Building an Object Detection Model from Scratch in Python,” [Online]. Available:

https://www.analyticsvidhya.com/blog/2018/06/understanding-building-object-detection-model-python/

[2] Yijing Yang, “EE569_Discussion_WK3_013120,” [Course Material]. Available for students in the course:

https://courses.uscden.net/d2l/le/content/17205/viewContent/298625/View?ou=17205

[3] Justin Liang, “Canny Edge Detection,” [Online]. Available:

http://justin-liang.com/tutorials/canny/

[4] Piotr Dollar and C. Lawrence Zitnick, “Fast Edge Detection Using Structured Forests”, [Online]. Available:

https://pdollar.github.io/files/papers/DollarPAMI15edges.pdf

[5] Samwoo Seong, “Development of Robot Movement Recognition System and Autism Diagnosis System,” [Online]. Available:

https://github.com/Samwoose/AutismDianosisSystem/blob/master/Project_Report_Autism_Robot.pdf

[6] C.-C. Jay Kuo, “EE 569: Homework #2,” [Course Material]. Available for students in the course:

https://courses.uscden.net/d2l/le/content/17205/viewContent/298314/View?ou=17205

[7] C. A. Bouman, “Digital Halftoning,” [Online]. Available:

https://engineering.purdue.edu/~bouman/ece637/notes/pdf/Halftoning.pdf