EE569- Introduction to Digital Image Processing

Homework #2

Sourabh J. Tirodkar
3589406164
Submission Date- 16 Feb 2020

Problem 1: Edge Detection

a) Sobel Edge Detector

1. ABSTRACT AND MOTIVATION

   Edge Detection is an image processing tool where we can extract an edge which is the difference in the pixel intensity. Edge distinguishes the object and the background. The process of edge detection is very important in analysis as it extract important features like lines, curves. By using Edge Detection as a tool, it can be applied to higher algorithms which includes Object Detection, segmentation, etc.

   The very basic method developed was Sobel Edge Detector.

2. APPROACH

   Sobel Edge Detector technique uses Intensity function and its 1st derivative to compute the edges of an image. There are 2 intensity functions each in horizontal and vertical directions. The first derivative is calculated and the local maxima, minima denotes the edges. It is basically drastic change in intensity value of pixel which is quite visible in the graph below. The first derivative calculation is known as intensity gradient.

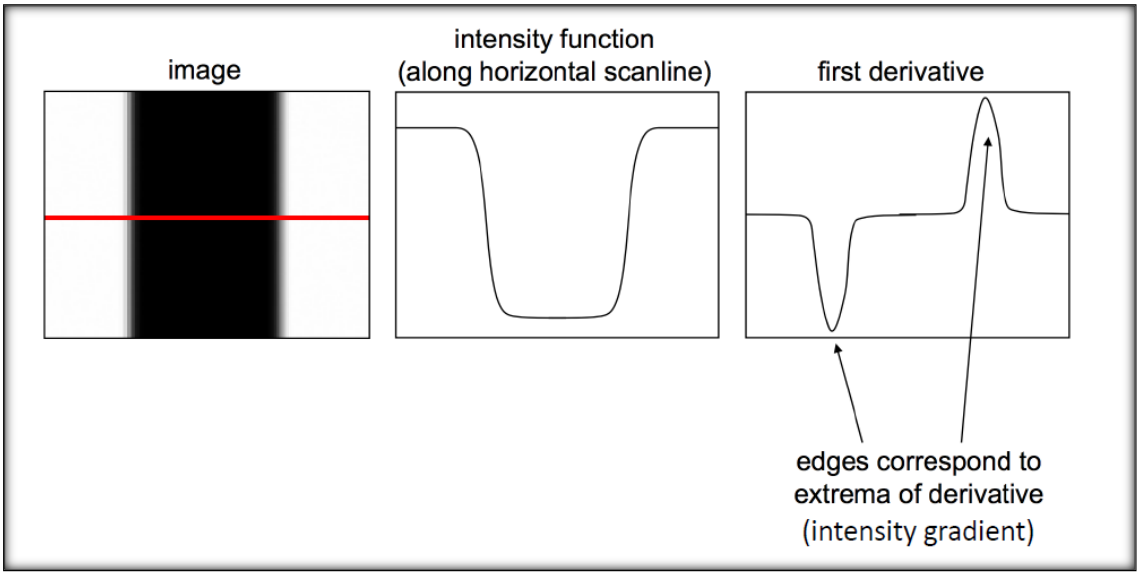


Fig 1. First Derivative graph which extracts edges

Gradient

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnitude

$$|\nabla f| = \sqrt{g_y^2 + g_x^2}$$

Orientation

$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

The gradient calculation is done using above formulas. We then convert continuous gradient to finite discrete form which is applied to the image.
I have used the following 3*3 filter masks which are called as Sobel 3*3 filter to extract the edges.

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

Gx

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

Gy

I applied the filter masks and processed convolution with the image that is provided (*Gallery.raw* and *Dogs.raw*). The filter mask is circulated throughout the whole image first by going left to right and then top to bottom.
Finally, I normalized the convoluted image after combining the X and Y gradients. The image pixels are inverted for visual purpose.

3. RESULTS

The following is the result after normalization of X and Y gradient:

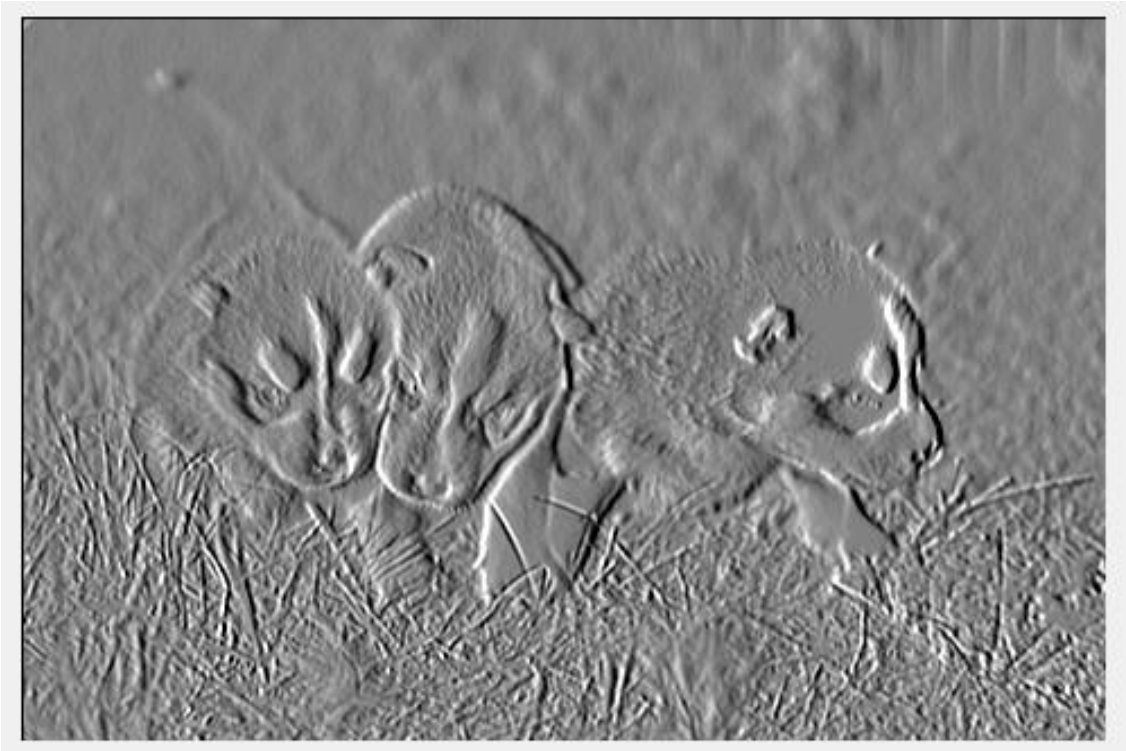


Fig.1 X Gradient of Dog Image

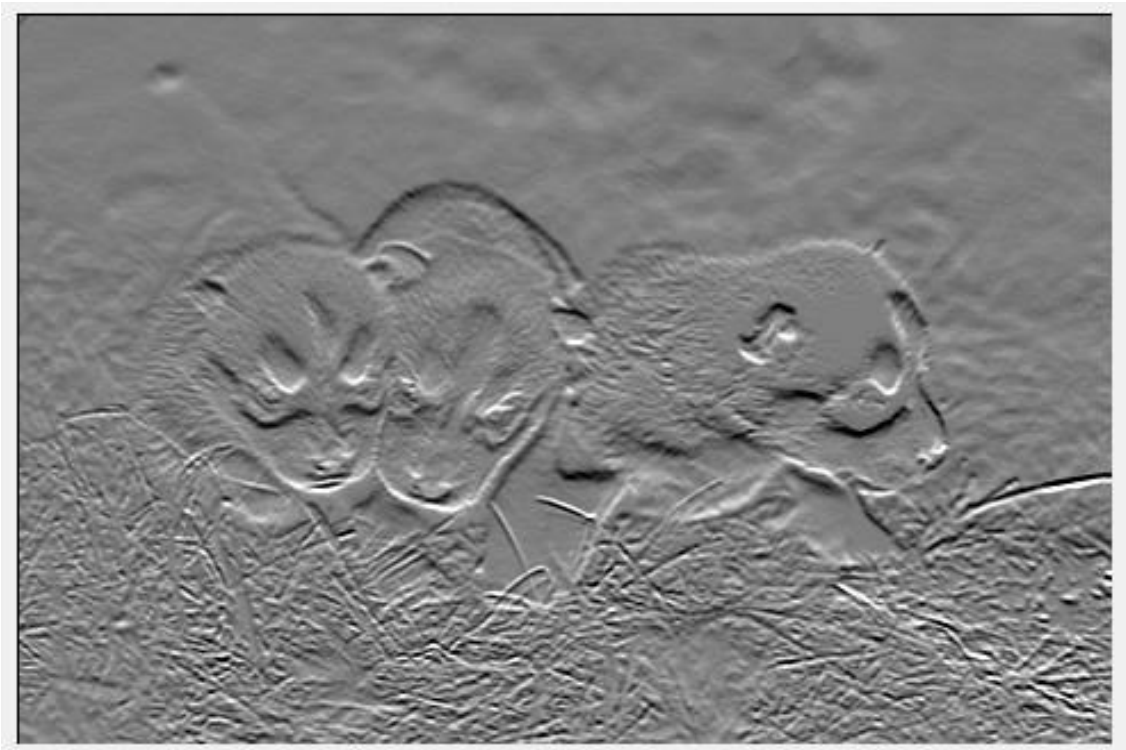


Fig.2 Y Gradient of Dog Image

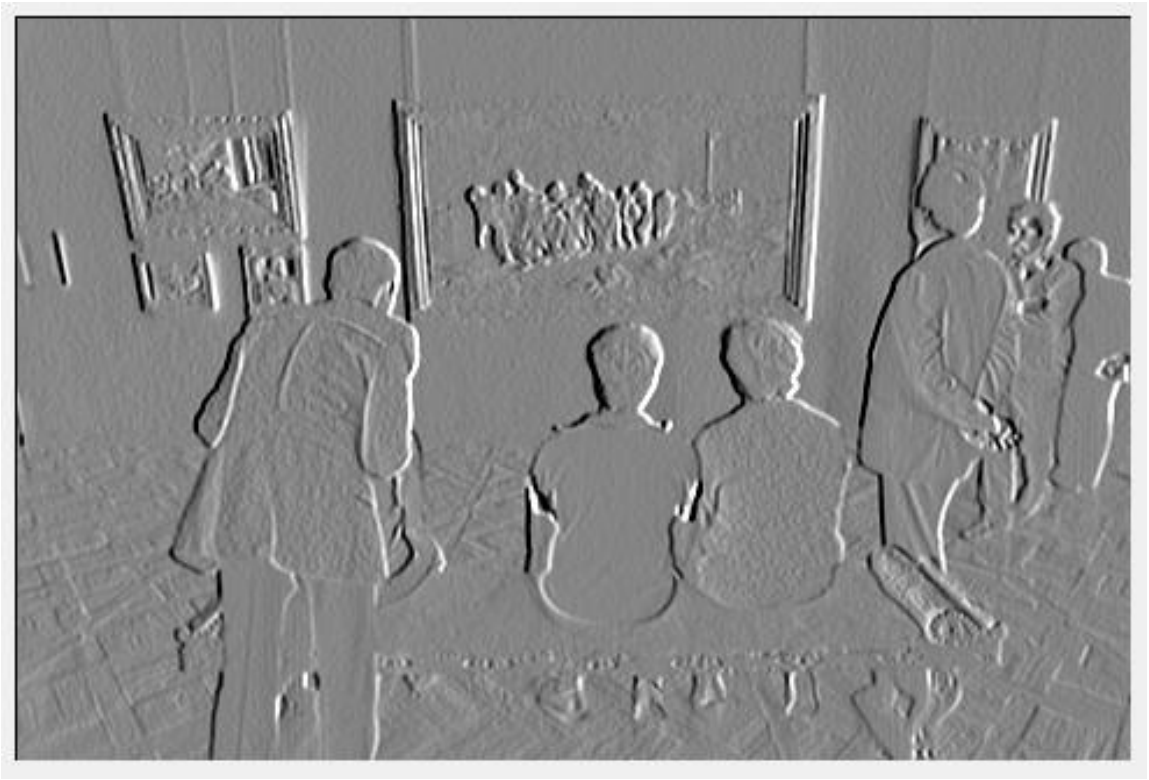


Fig.3 X Gradient of Gallery Image

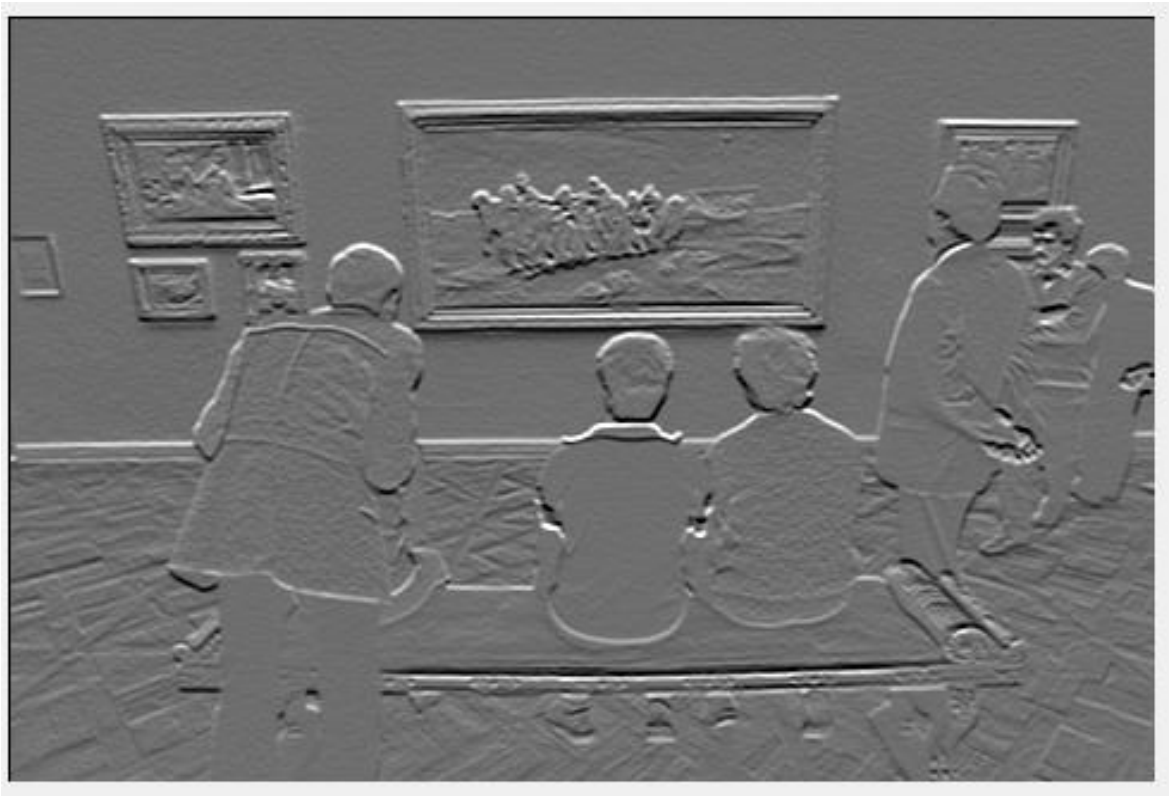


Fig.4 Y Gradient of Gallery Image

The normalized gradient magnitude is shown as follows:

Fig.5 Gradient Magnitude of Dog Image

Fig.6 Gradient Magnitude of Gallery Image

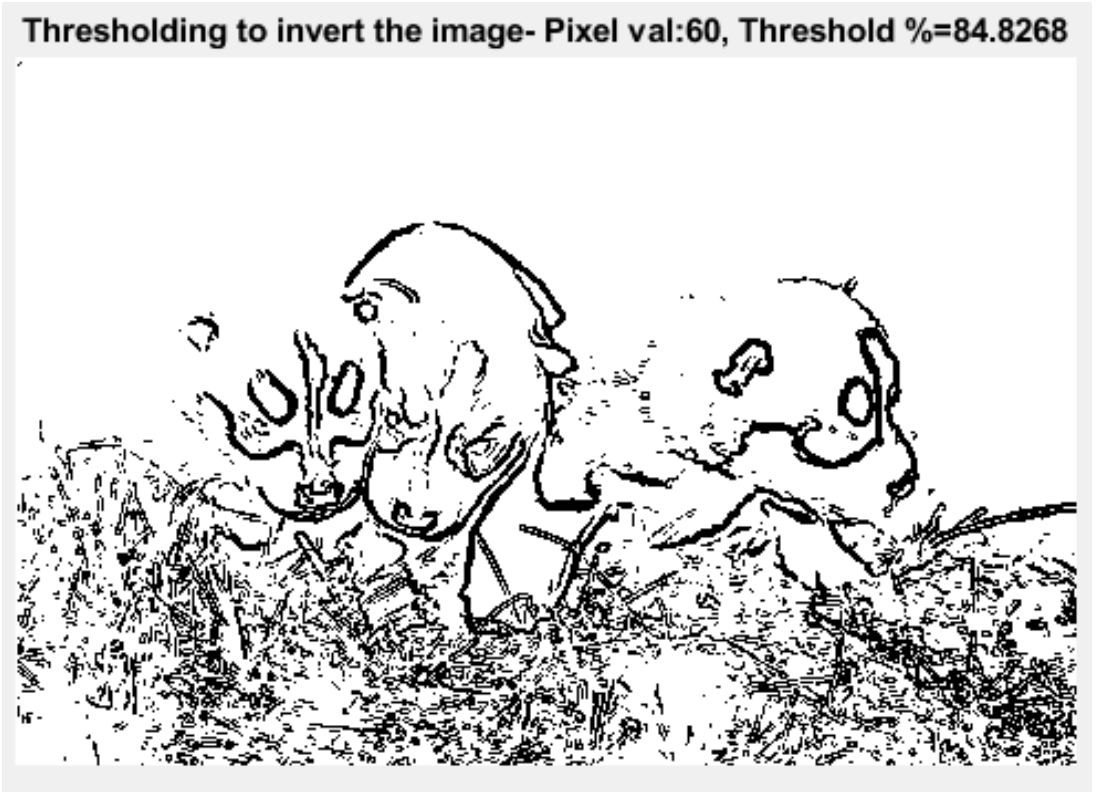


Fig.7 Thresholding to Invert the Dog Image

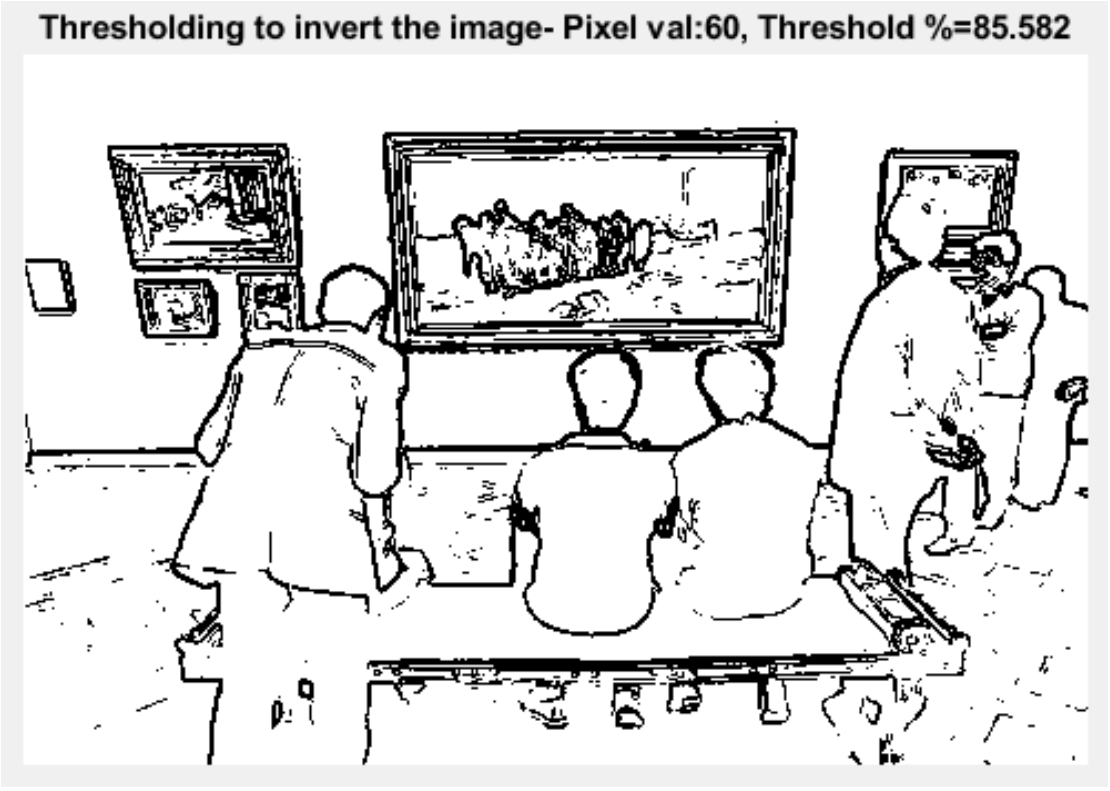


Fig.8 Thresholding to Invert the Gallery Image

We can give threshold value in Sobel Edge Detector which kind of effects the final output of an image. The more the threshold (90%), the lighter edges do fade off. Considering threshold of 85%, we can see edges but the image seems to be dark sometimes.

4. DISCUSSION

As we can see, Sobel Edge Detector algorithm is very simple to apply. It does detect the edges and orientations are proper as well.

The disadvantages being it is sensitive to noise and also reconsidering the existing edges.

Overall, Sobel Edge Detector cannot produce accurate edge detection which are thin and smooth edge.

The 2nd derivative of intensity function yields better results, and is called Laplacian of Gaussian.

b) Canny Edge Detector

1. ABSTRACT AND MOTIVATION

After development of Sobel Edge Detector, and the results of edge detection were not at its best, so Canny Edge Detector was introduced. The main purpose of this was to being insensitive to noise and also not considering the existing edges as in Sobel. It uses image intensity value and Non-Maximum Suppression technique with double thresholding.

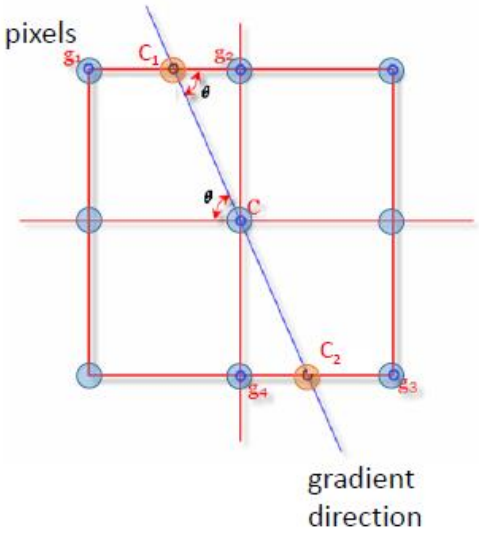
Also, the motivation to develop Canny was to thin the edges as the edges were too thick in Sobel operation. This is known as NMS which gives thin edges (candidate edges)

2. APPROACH

The basic steps followed in Canny edge detector are as follows:

a. Filter Image with Gaussian
b. Finding Intensity gradients of an image
c. Non-Maximum Suppression (NMS)
d. Double thresholding (hysteresis thresholding).

NMS Approach:

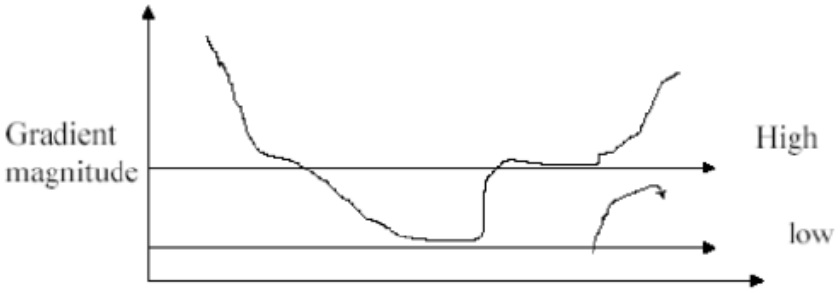


We suppress all gradient values to 0 except the local maxima. This approach leads to candidate edges. We need to move in 0°, +45°, 90°, -45° direction and then apply algorithm.

We compare edge strength of current pixel with edge strength in positive and negative directions. Then, second comparison is between the edge pixels with those in the masks. If it

is greater then pixel value stored, If not it will be suppressed. This leads to thin edges as said earlier.

Hysteresis thresholding:



It used 3 regions to determine an edge. We define 2 thresholds say high and low.

a. If the intensity gradient is more than high threshold, then it is considered as an edge.
b. If the gradient lies below the low threshold, then it is discarded and it is not an edge.
c. The third case, where the gradient lies in between 2 thresholds, then it is considered as an edge if its neighbouring pixel is an edge.

3. RESULTS

I have used open source provided by MATLAB for the implementation of Canny Edge Detector. The result of Canny Edge Detector are as follows:

Fig.9 Canny Edge Detector with 1st threshold selection (dog)

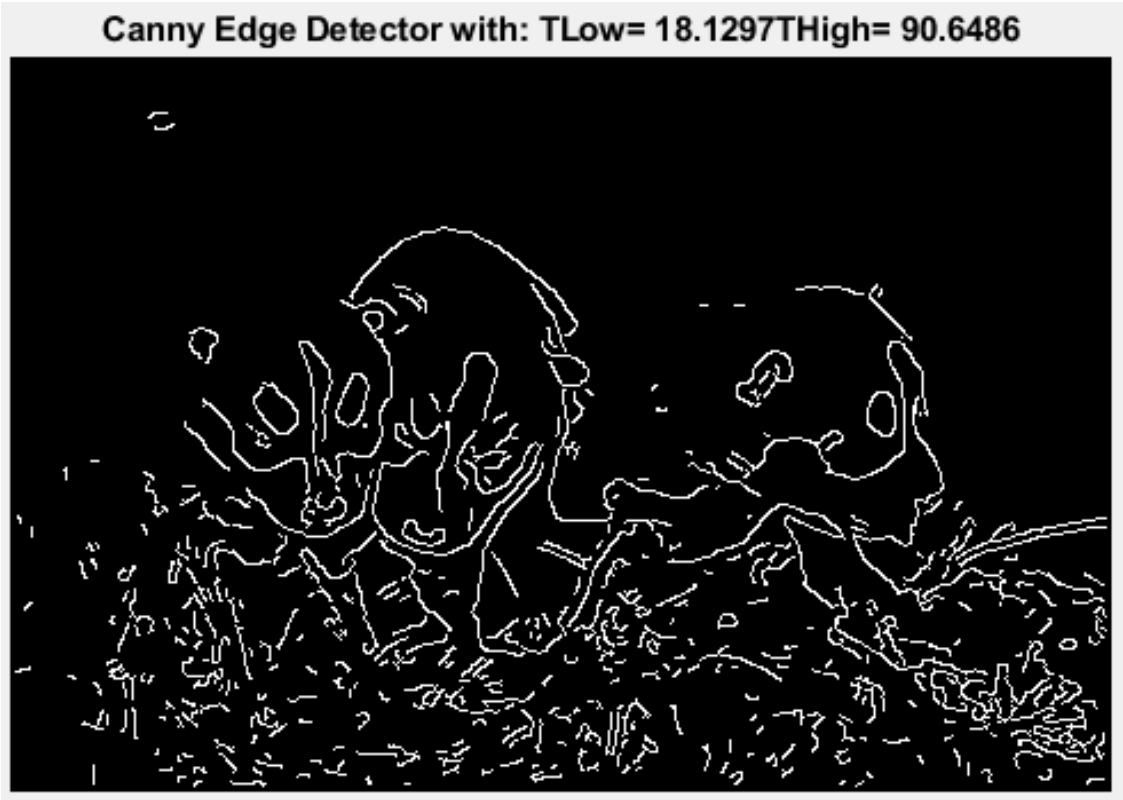


Fig.10 Canny Edge Detector with 2nd threshold selection (dog)

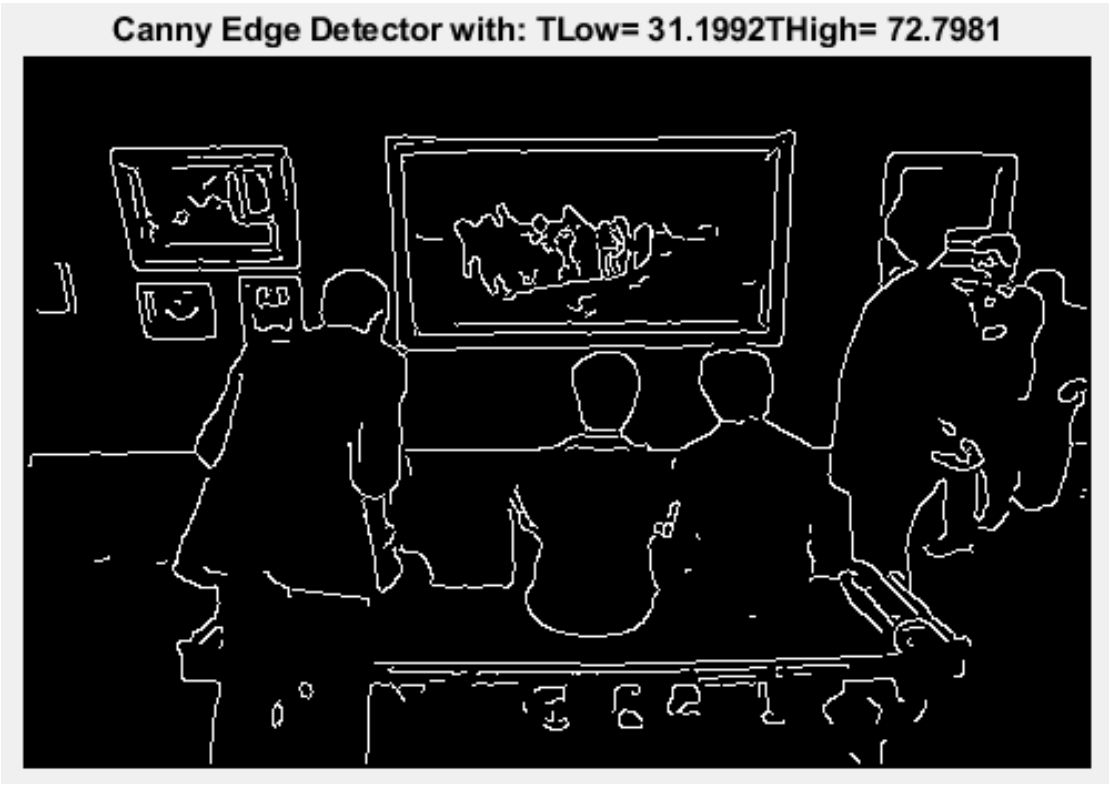


Fig.11 Canny Edge Detector with 1st threshold selection (gallery)

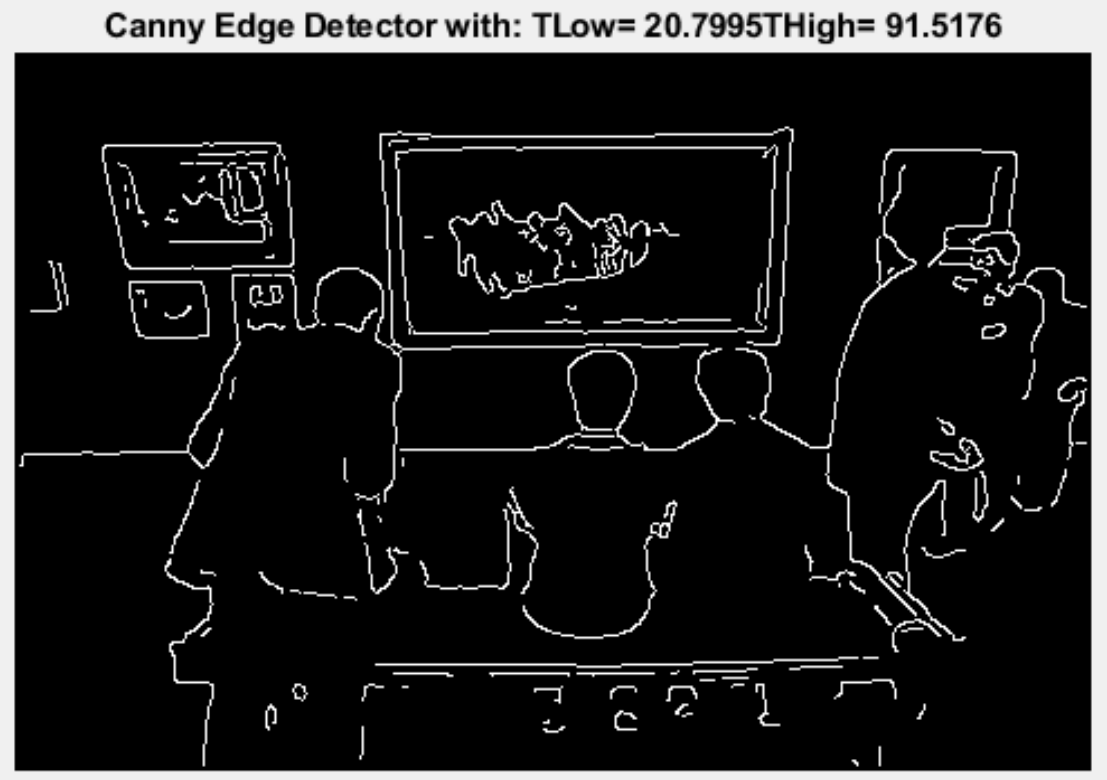


Fig.12 Canny Edge Detector with 2nd threshold selection (gallery)

The threshold values used are as follows:

'*Dog.raw*'
1st Case:
T_low=27.19 T_high=63.35
2nd Case:
T_low=18.13 T_high=90.64

'*Gallery.raw*'
1st Case:
T_low=31.19 T_high=72.19
2nd Case:
T_low=20.79 T_high=91.51

4. DISCUSSION

The selection of thresholds values in Canny becomes so important. If a threshold is set too high then it will miss the image information. If the threshold is too low, then it will creep in the noise. Comparing, fig 9 and 10, we can see that edges(information) in fig 10 have been disappeared. So, threshold selection becomes very crucial in canny edge detector.
The results are better than the Sobel Edge Detector mechanism but there are stills ways to improve the results.

c) Structured Edge

1. ABSTRACT AND MOTIVATION

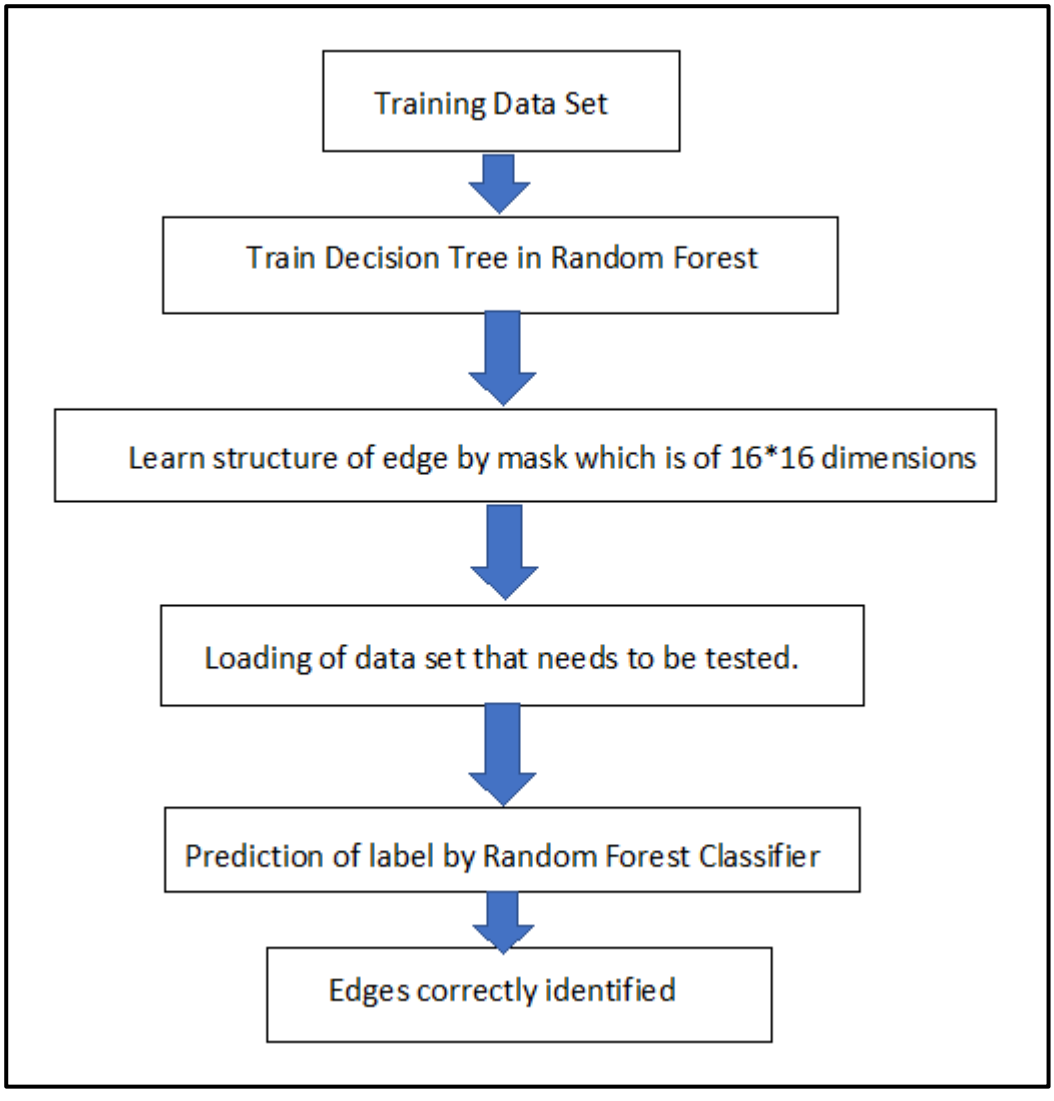
   Structured Edge (SE) is a machine learning based method to detect an edge from the image. It is based on the fact that each edge has a structure. There are traditional approaches but they failed to detect an edge to the fullest. So, there was need to detect an edge with setting multiple parameters and then detecting an edge.

2. APPROACH

   We use Random Forest for in the algorithm of Structured Edge. It has many decision trees which are trained using many random samples. A 16*16 mask is used for prediction of edge or not. We also need to pass the parameters value in the function. These parameters play an important role as the edge detection map changes due to it. Those are the following: nms (no maximum suppression), nTreesEval, etc.

   After getting probability edge we convert it into binary edge map using a threshold value.

Flowchart for Structured Edge



3. RESULTS

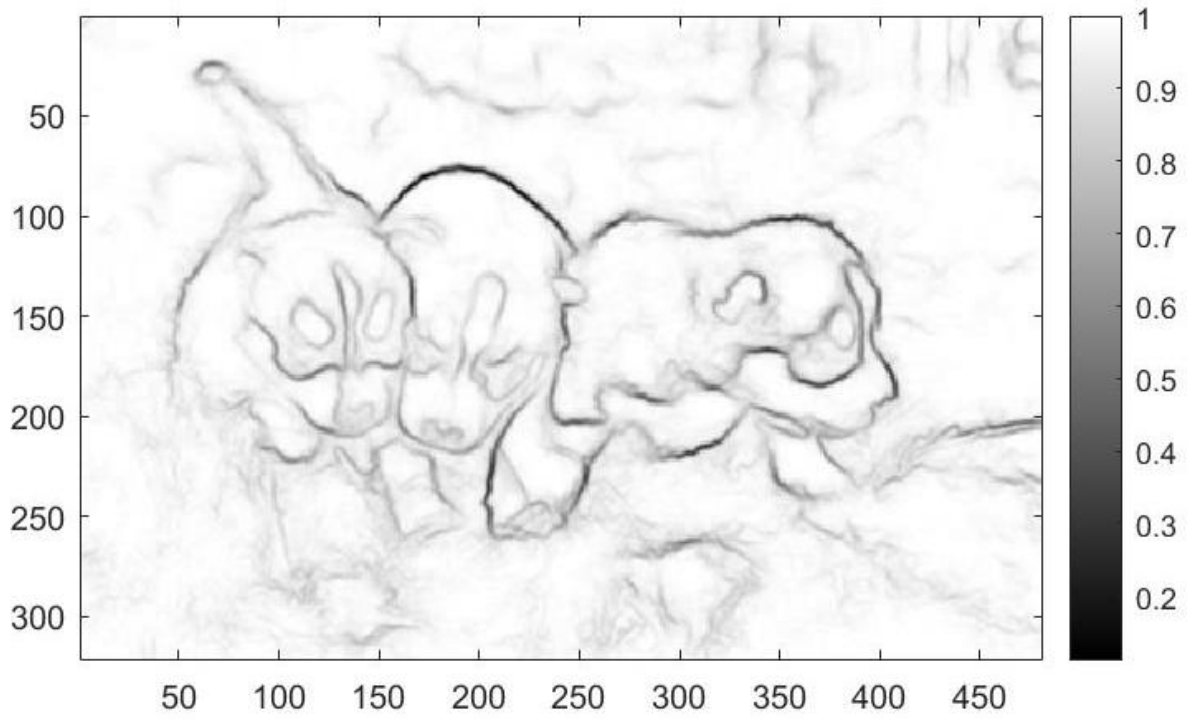


Fig.13. Edge Detection of Dog using SE

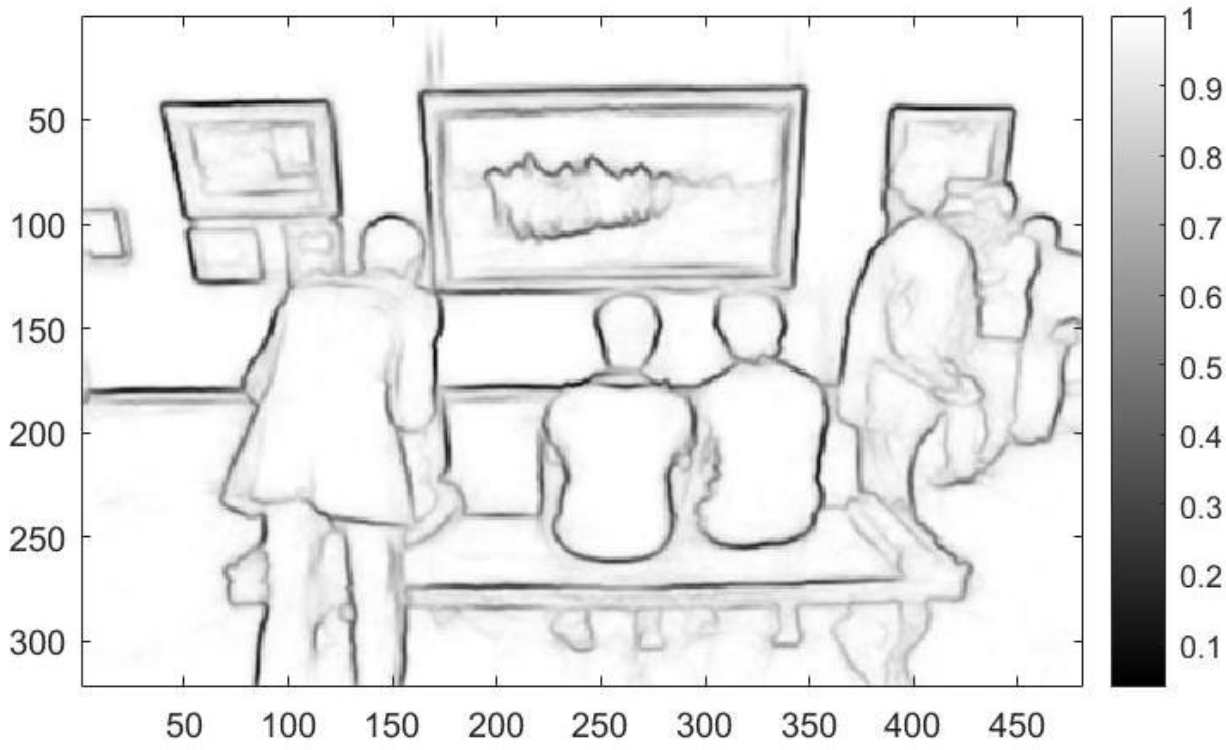


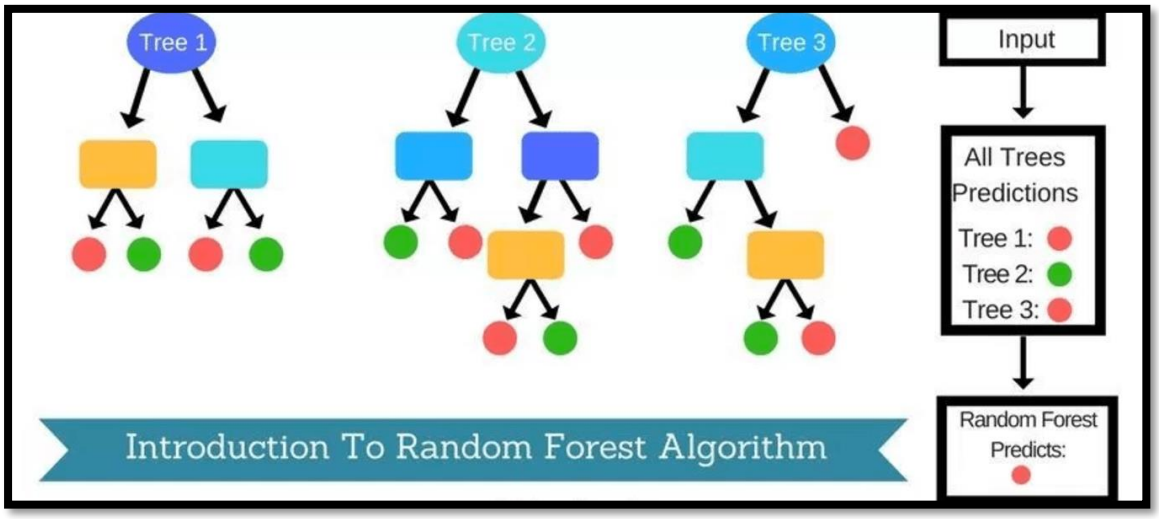
Fig 14. Edge Detection of Gallery using SE

4. DISCUSSION

Random Forest:

Random Forest is a learning method used for classification, regression and many other operations. The decision tree is basic element that builds the random forest. Random Forest does include many decisions trees. Each output of decision tree accounts for a class in classification algorithm.

A decision tree consists of Nodes (Variables for testing), branches (test results), leaves (classification).



Construction of Decision Tree:

a. A decision tree f(x) classifies sample x by sending them in either in branch 1 or 2 and stop when a leaf node is reached.
b. Highest priority branch is considered while choosing the features.
c. The parameter (y) is compared with threshold, then the decision is made whether to follow branch 1 or 2.
d. Classification ends when maximum depth is reached.
e. We calculate gain using entropy.
f. The output of tree is stored in the leaf node with the target class.

RF Classifier:

a. Random Forest is an assemble of many decision trees.
b. Output Predictions from number of decision trees are merged together to form a single output.

The results of SE are shown above which is then compared with the results of Canny.

Chosen Parameters in SE:

a. nTreesEval=1
b. nms=1
c. multiscale=1
d. sharpen=1

Setting of nms is important as it does nan maximum suppression after edge is detected and thus resulting into thin edges. If nm=0 then it accounts for thick edge. Sharp=1 makes the edge looks sharper. Multiscale is used for accuracy. nTreesEval=1 results in faster results.

Comparison of Canny vs Structured Edge:

a. Canny Edge detector has thick edges compared to Structured Edge.
b. The edge information is pronounced more in SE rather than Canny.
c. The edge information may lose in Canny.
d. Canny uses Gaussian derivative to detect an edge while SE uses multiple parameters to detect the same.
e. Thus, SE performs better than Canny.

d) Performance Evaluation

1. ABSTRACT AND MOTIVATION

   Performance Evaluation is a measure to evaluate the various edge detectors. This is necessary as we need a check which method detects the edges in a better way. In performance evaluation, we need to generate machine contours which should be better than human provided contours. As humans generated map can be of various types, it is important we have machine generated maps.

2. APPROACH

   As discussed above, humans binary map (ground truths) can be of various types, it is necessary to take mean of certain performance measure, for eg- mean precision, mean recall. We need to find error in the map. Every pixel will lie in the following 4 categories:

   a. True Positive
      These are the pixels in edge map which coincides with pixels in the ground truth. The algorithm is successful in detecting these edge pixels.
   b. True Negative
      These are non-edge pixels in edge map with coincides with non-edge of ground truth. The algorithm is successful in detecting these non-edge pixels.
   c. False Positive
      Each pixel in edge map correspond to non-edge in ground truth. The algorithm fails in this case.
   d. False Negative
      Each non edge pixel in edge map corresponds to edge pixel in ground truth. The algorithm fails in this case.

   The performance can be evaluated with a parameter called F measure which is a function of precision and recall.

$$\text{Precision}: P = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}}$$

$$\text{Recall}: R = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R}$$

If we make precision higher then it makes recall lower. Ideally, we need both to be appropriate and hence quantity F measure is important. Higher the value of F measure, the better edge detector it is.

Steps to be followed:

a. Run edgesEvalImg() function. We need to provide ground truth file along with edge map.
b. Output parameters like thrs, cntR, sumR, cntP, sump, V will be displayed.
c. Using these parameters;
   P = cntP/ sumP
   R = cntR/ sumR
d. When P and R is obtained, then we find F measure by
   F = 2*P*R/ (P+R)

3. RESULTS

Image	Detector	Precision	Recall	F score
Dog	**Sobel**	0.25	0.75	**0.378**
Dog	**Canny**	0.73	0.53	**0.621**
Dog	**SE**	0.51	0.87	**0.648**

Table 1. Dog Image: P, R and F for all 3 algos

Image	Detector	Precision	Recall	F score
Gallery	**Sobel**	0.64	0.88	**0.74**
Gallery	**Canny**	0.81	0.51	**0.78**
Gallery	**SE**	0.87	0.79	**0.821**

Table 2. Gallery Image: P, R and F for all 3 algos

Ground Truths **(DOGS)**	Sobel			Canny			SE		
	P	R	F	P	R	F	P	R	F
GT1	0.06	0.79	0.12	0.2	0.73	0.32	0.31	0.87	0.46
GT2	0.1	0.5	0.16	0.32	0.54	0.41	0.41	0.55	0.47
GT3	0.1	0.5	0.16	0.32	0.54	0.41	0.41	0.55	0.47
GT4	0.1	0.5	0.16	0.32	0.54	0.41	0.41	0.55	0.47
GT5	0.1	0.2	0.16	0.32	0.54	0.41	0.41	0.55	0.47

Table 3. Dog Image: P, R and F for all ground truths

Ground Truths **(GALLERY)**	Sobel			Canny			SE		
	P	R	F	P	R	F	P	R	F
GT1	0.38	0.92	0.54	0.61	0.77	0.67	0.58	0.95	0.72
GT2	0.34	0.93	0.51	0.56	0.81	0.66	0.52	0.95	0.67
GT3	0.36	0.91	0.52	0.56	0.77	0.65	0.55	0.95	0.70
GT4	0.33	0.92	0.49	0.57	0.83	0.68	0.52	0.97	0.68
GT5	0.51	0.92	0.65	0.78	0.77	0.78	0.75	0.93	0.83

Table 4. Gallery Image: P, R and F for all ground truths

4. DISCUSSION

a. The table pf precision, mean and F score for each ground truth is given above.

Performance of different edge detectors:

Sobel Edge Detection

As seen from the table, we can see that Sobel has very less F score than Canny and SE. It is the weakest algorithm from the rest. As seen, it has thick edges. For better quality of edge, we need to make those thin.

Canny Edge Detection:

It is better than the Sobel as it uses NMS and Hysteresis thresholding which reduces the edge width.

Structured Edge:

SE algorithm is supposed to be best in edge detection. The edges are thinner than Sobel and Canny.

b. F measure is a quantity which is image dependent. The 'Gallery' image will get higher F score no matter which algorithm is used. The precision is also achieved higher in 'Gallery' than 'Dog' image. The recall parameter which is used in calculating F score can be varied. As the dog image has lot more edges (eg- grass, background) the F measure is lower than gallery image. The edges in dog image can be considered as noise which accounts for lower F score.

c. F measure is a quantity which is dependent on precision (P) and recall (R). Varying this value can alter F score. It is a value which takes both P and R into account and adjust F accordingly.

No, it is not possible to achieve higher F score if precision (P) is significantly higher than recall (R).

If sum of P and R is constant say C, so P+R= C then R= C- P;

$$\text{So, } F = \frac{2*P*R}{P+R} = \frac{2*P*(C-P)}{C} < C$$

When R=P,

$$F = \frac{2*P*P}{P+P} = P$$

It does achieve maximum value when Precision (P) = Recall (R)

Problem 2: Digital Half-Toning

a) Dithering

1. ABSTRACT AND MOTIVATION

In modern world, we aim to save cost of whatever products we can. For an image printing, we have Grey Scale (8 bit) or RGB Scale (24 bit). There's high cost involved in printing this. So, a process of HALF TONNING is required. It is a process where a digital image is transformed into binary scale (black or white) for the printing purpose. As the output of Half toning is binary, it does reduce cost to greater extent where only black ink is used for printing. Most printers cannot reproduce different shades of grey or different colour combinations in RGB image. Digital Half thus makes it easier to print those images in just 1 ink.

One of the methods to do Half Toning is DITHERING.

2. APPROACH

Dithering is the simplest method to do digital Half Toning. It considers a threshold value and all the image pixel one by one are compared to this threshold value (T). If the pixel value is less than T, then give it a value 0 (black) an if this condition is not satisfied then give it value 255 (white). The deciding factor is T. We need mathematical approach to find its value.

Dithering has 3 methods:

1. Fixed Thresholding
2. Random Thresholding
3. Dithering Matrix

Fixed Thresholding Approach:

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

Here, we need to select threshold (T). I have selected T= 128 to get better results.

Random Thresholding Approach:

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

Here, I have generated random number using inbuilt function 'rand' which is of the same dimensions as of image. The random number are made it in range from 0 to 255.This modified random number is considered to be threshold. The image pixels are compared pixel by pixel to this threshold and above operation is followed.

Dithering Matrix Approach:

In this method, we have to use dithering matrix. The order of the matrix is up to us. The 2*2 dithering matrix is given as:

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

Using this we can generate higher order dithering matrix by following equations:

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

Now, next step is to generate threshold matrix using the dithering matrix, which is calculated as:

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

Using this threshold, as said, we need to compare every pixel and then make it to either black or white.

3. RESULTS

Fig. 15. Light House Original image

Fig. 16. Light House Fixed Thresholding Image

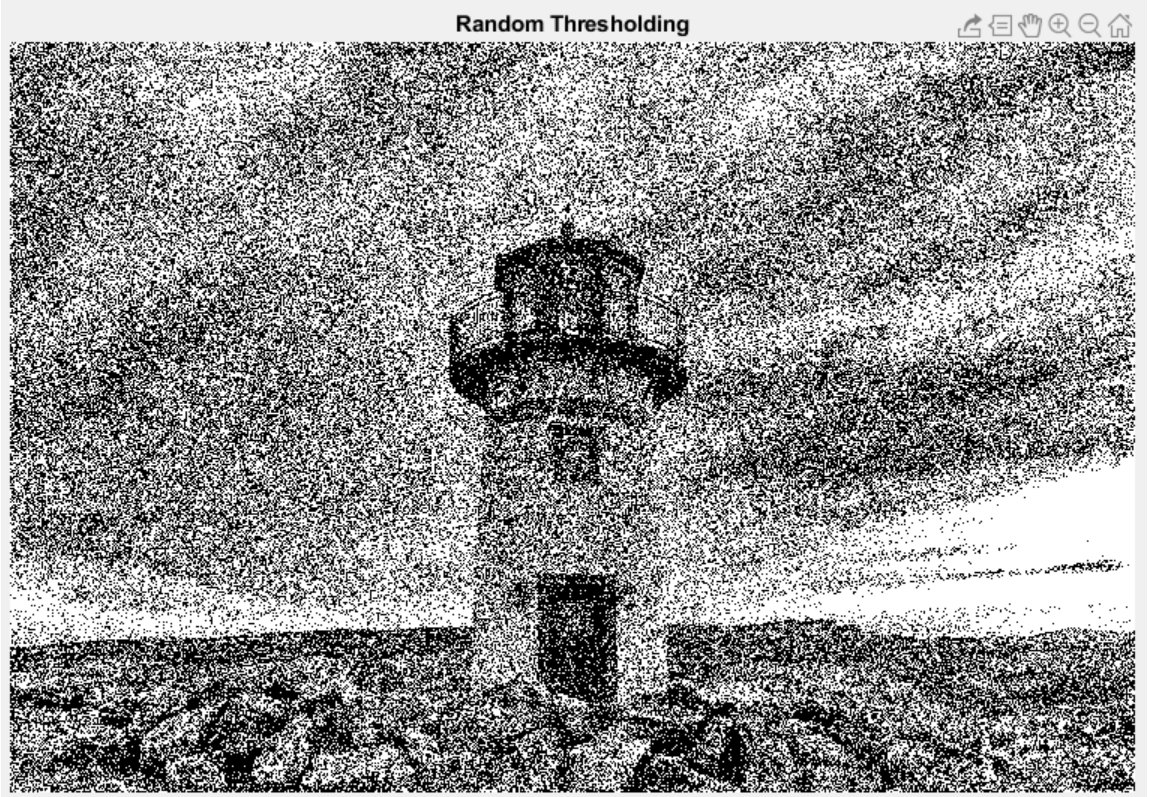


Fig. 17. Light House Random Thresholding Image

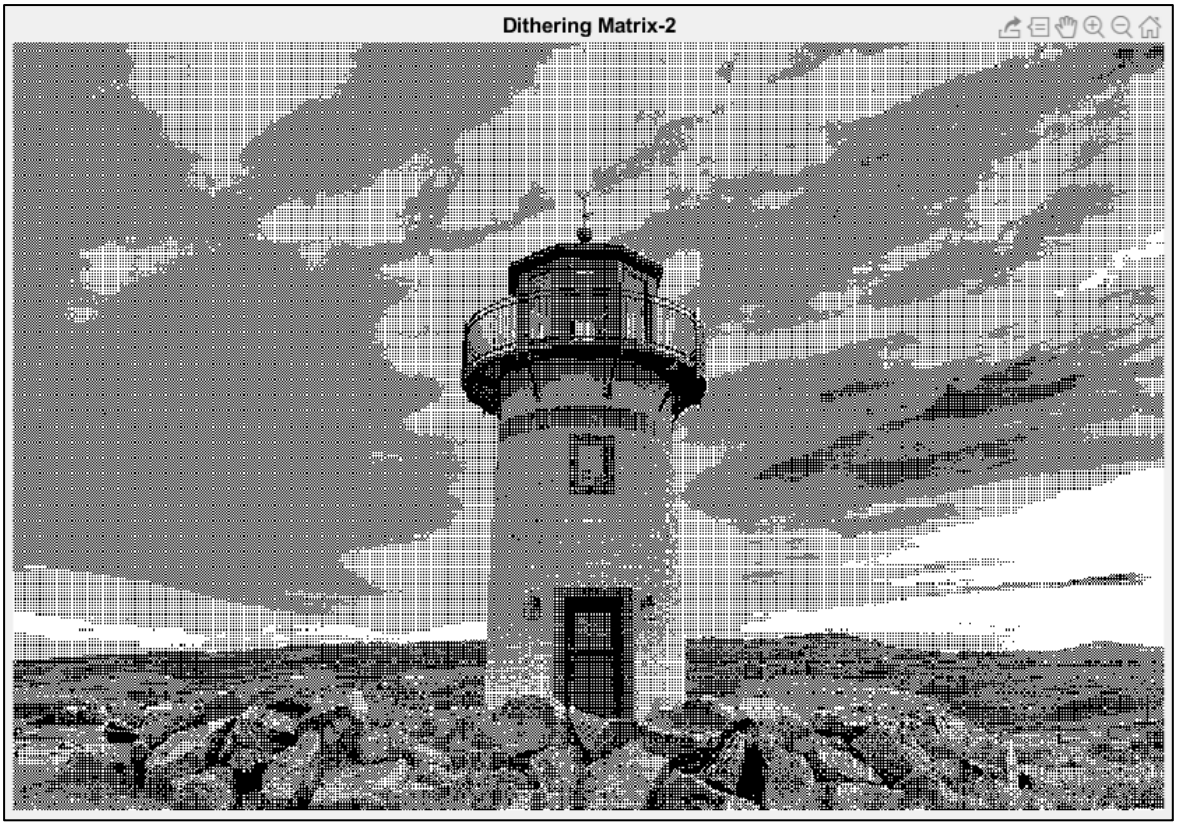


Fig. 18. Using Dithering Matrix-2 as Thresholding

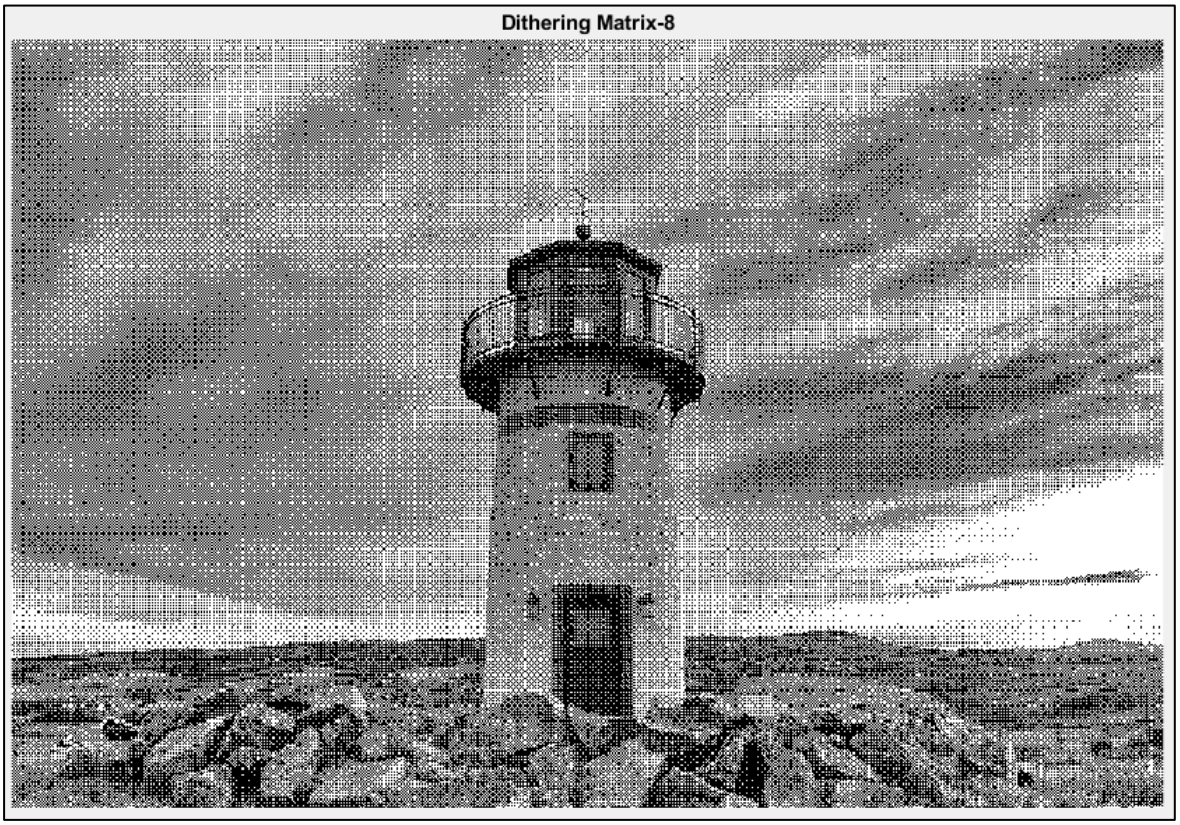


Fig. 19. Using Dithering Matrix-8 as Thresholding

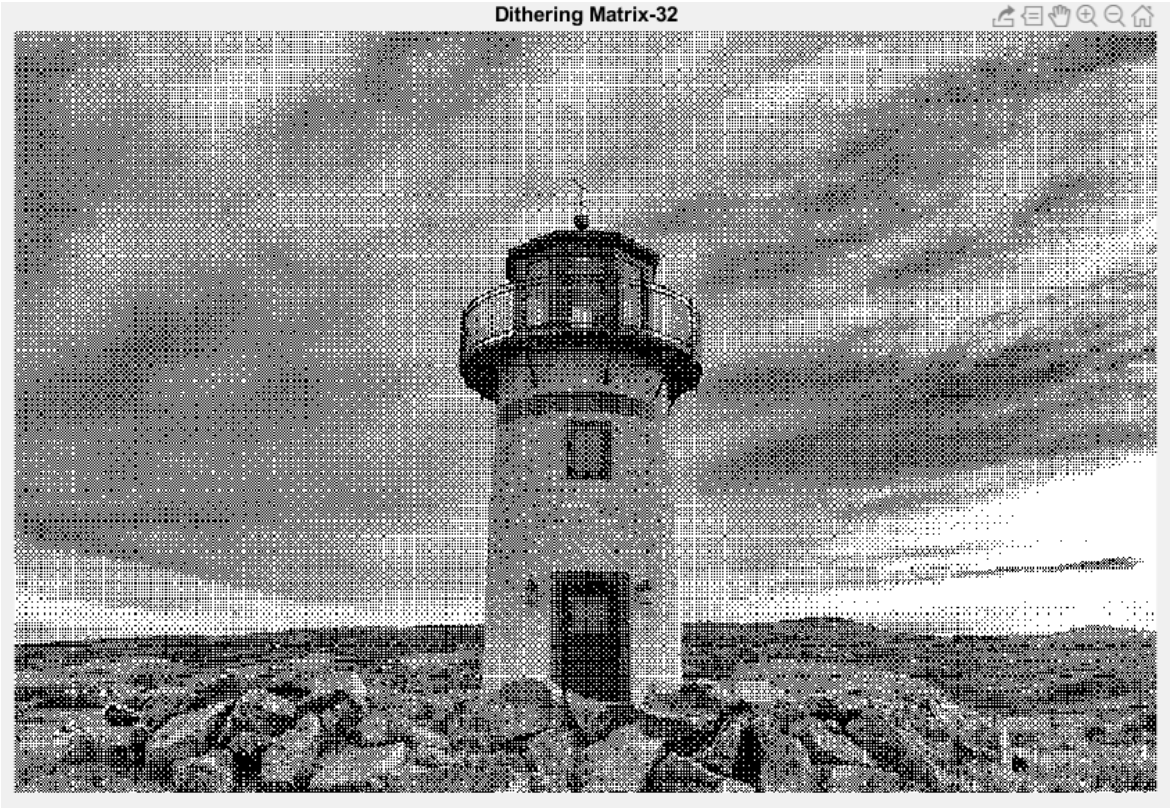


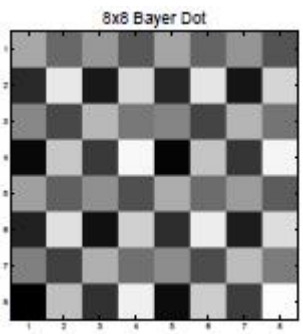
Fig. 20. Using Dithering Matrix-32 as Thresholding

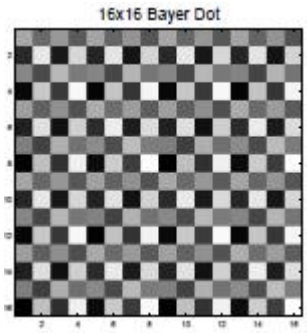
I have implemented Digital half toning using dithering process. As said above, it has 3 different approaches, each one of its output is plotted.

Fig 15 is the original image of Light House. Fig 16 is the output of Fixed thresholding where T=128. The binary scale is clearly visible in it.
Fig 17 is the result of Random Thresholding. The output does appear to be like added noise, but upon zooming it, we can see the binary output.
Fig 18, 19, 20 shows the output of half toning by dithering matrix method of n= 2, 8, 32 respectively.

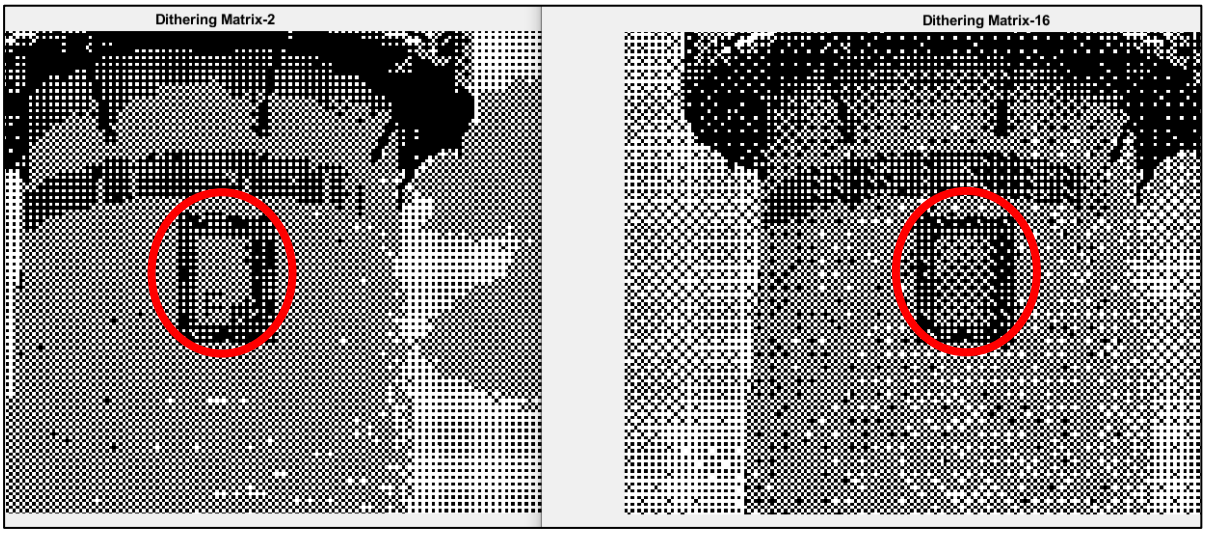
4. DISCUSSION

The output of fixed thresholding is a binary image but we get a continuous monotone in that process. Thus, quality of image is poor.
In order to avoid those disadvantages, we use random thresholding, which gives a better result as shown in above figure.
The better results are obtained by using dithering matrix method. It creates a perception of continuous grey levels in an image. The values in the matrix gives exactly how likely the dot is to be turned on. The 0 value indicates it will be turned on most likely while 3 indicates it is least likely.
As order of dithering matrix gets increased, a greater number of patterns are generated. Eg- For 4*4, there are 17 patterns generated. A finer amplitude quantization over larger area and good rendition over smaller area is obtained as we used Bayer pattern of high order
More the number of patterns better is the visual quality of image.





The images after zooming in different matrix is shown below:



Higher order Dispersed Dot Screening using Bayer matrix has many advantages over other two and so it is preferred in most cases.
Some of properties are as follows:

a. No trade-off between resolution and grey levels.
b. Transitions between texture are more visible.

We can compute PSNR to compare each of the following implementation.

b) Error Diffusion

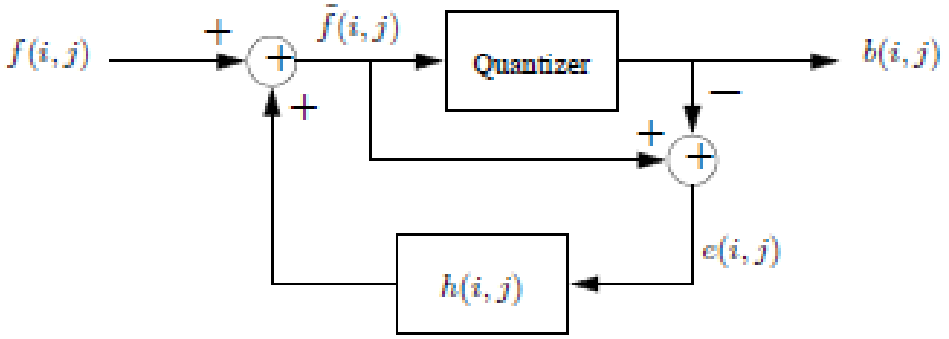
1. ABSTRACT AND MOTIVATION

   Error Diffusion is the other method to do half toning process. If there is a text in an image, when applied dithering, it is unable to read. So, a reformed method called error diffusion was used. Unlike dithering where we mean only to specific pixel, in error diffusion, we propagate error all the way to all image pixels which gives enhanced edges and better visual quality.

2. APPROACH

   In Error Diffusion as the name suggest, main aim is to diffuse the error to the neighbouring pixels. I first made the copy of the image to do binarizing. For this, a threshold is chosen to give value 0 or 255. The next step is calculating error and also diffusing it to neighbouring pixels. Error is given by e=image_copy- binarized_image

Filter View of Error Diffusion

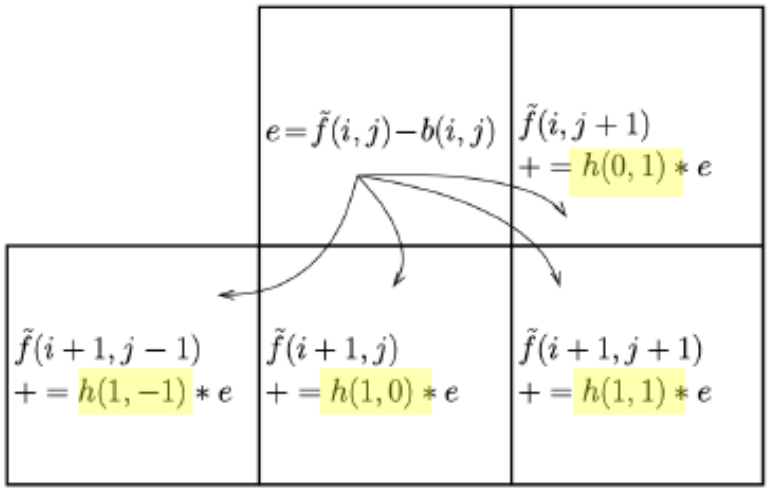


$$b(i,j) = \begin{cases} 255 & \text{if } \tilde{f}(i,j) > T \\ 0 & \text{otherwise} \end{cases}$$

$$e(i,j) = \tilde{f}(i,j) - b(i,j)$$

$$\tilde{f}(i,j) = f(i,j) + \sum_{k,l \in S} h(k,l)e(i-k, j-l)$$

Diffuse error forward using the following scheme

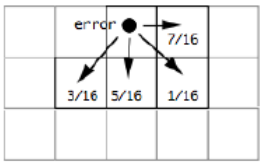


It has different choices of h to be choose from. The matrix for each is given below:

Different choices for *h* -- Error diffusion matrix

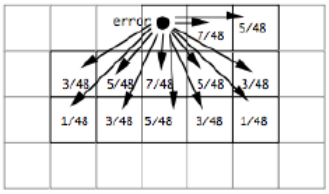
- **Floyd-Steinberg**

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



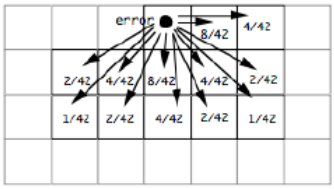
- **JJN**

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



- **Stucki**

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



Also, scanning order becomes important as we will scan the pixels and diffuse the error. For this problem, we will be using Serpentine Scanning as advised.

3. RESULTS

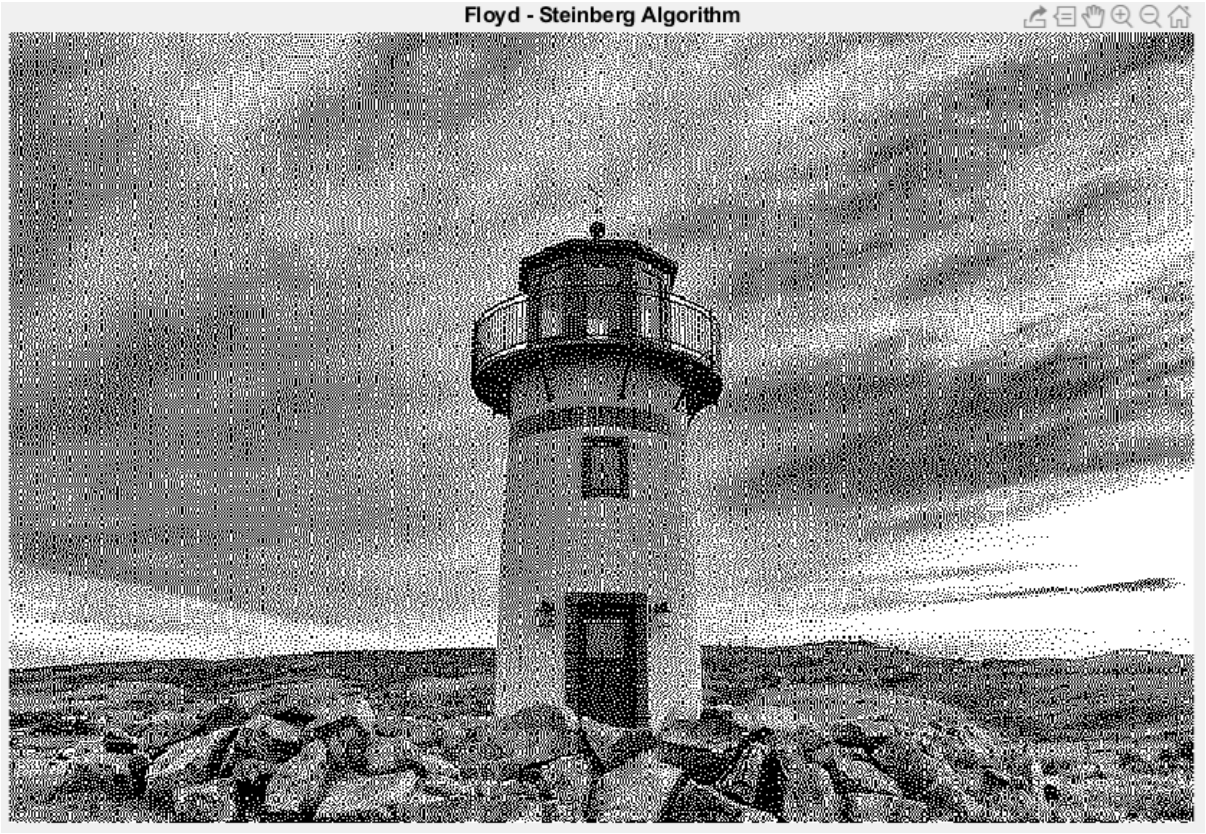


Fig. 21 Floyd Steinberg Implementation

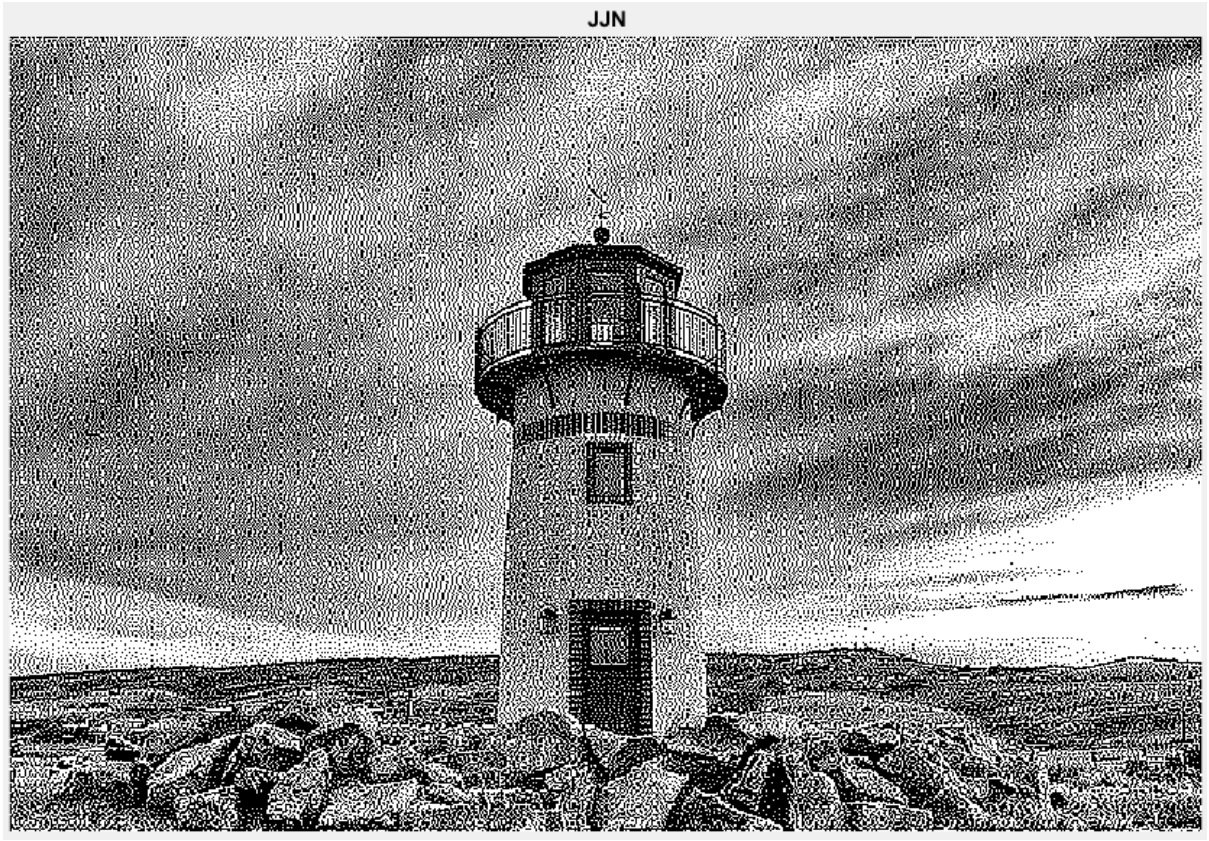


Fig. 22 Jarvis Implementation

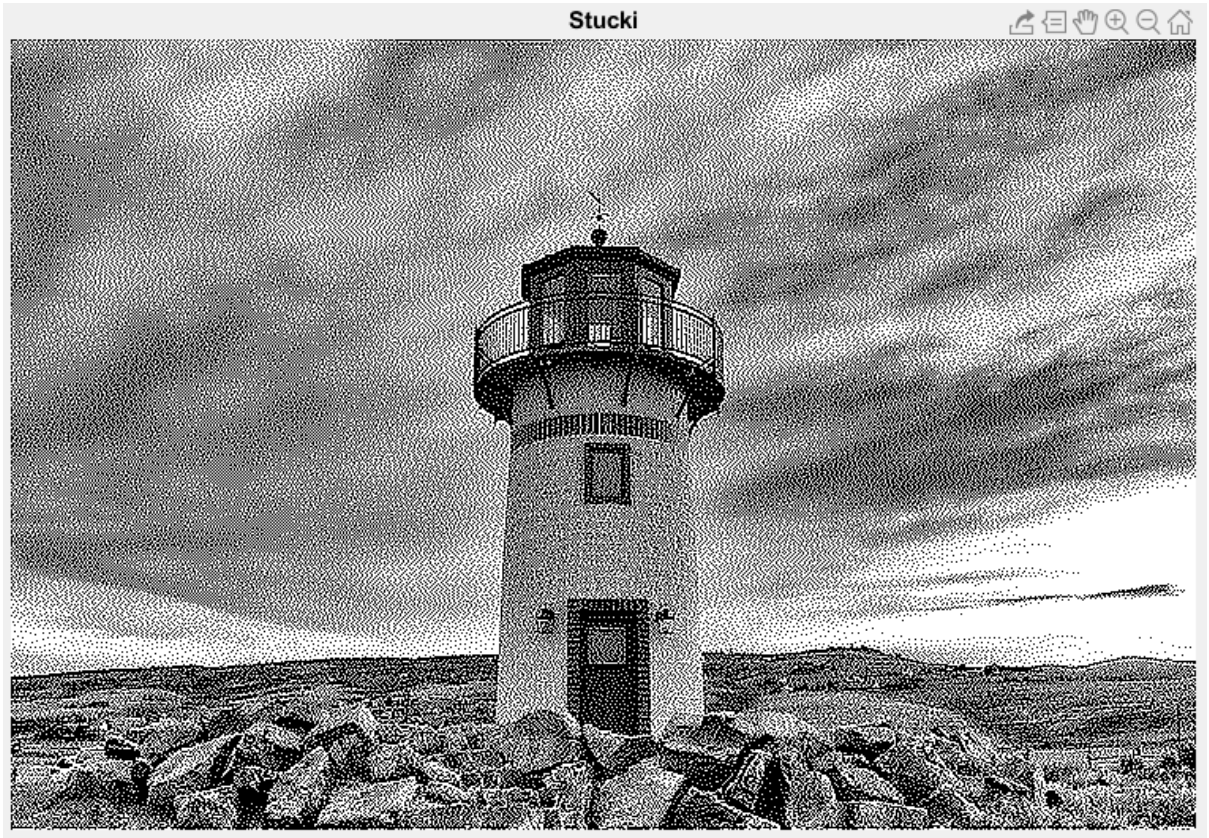


Fig. 23. Stucki Implementation

4. DISCUSSION

Comparing between Floyd-Steinberg, JJN and Stucki:
I feel 3*3 matrix works better than 5*5 matrix. So, Floyd-Steinberg is better than both JJN and Stucki algorithms. Also, between Jarvis and Stucki, there is hard to find any difference in the image. I could brightness difference between JJN and Stucki. We used Serpentine scanning for the implementations. The result of Serpentine scanning is better than Raster scanning.

As seen from the above figures in result section, we can definitely comment on the quality which is better in Error Diffusion rather than Dithering. It is obvious that dithering process uses pointwise operation which is why quality deteriorates. The neighbourhood pixels in an image are of much importance and hence propagating error gives enhanced and better visual quality image. Artifacts are avoided in Error Diffusion.

Own Idea:
I think working on Error Diffusion method would yield better quality images. I would probably say to implement a new Error Diffusion matrix which will have better perception of how the dots will be turned on step by step. Also, I feel a 3*3 error diffusion matrix would fetch me improved quality of image. As matrix dimensions increases, the quality degrades.
Adding, checking the matrix of Floyd, JJN and Jarvis, it can be seen that more error is distributed in the diagonals which causes artifacts. If less error is disbursed along diagonals, then image quality would improve. I would rather prefer Serpentine scanning.

c) Color Half-toning with Error Diffusion

1. ABSTRACT AND MOTIVATION
As seen from all the abstracts above, half toning is important and we have performed only on Grey scale images. It does reduce 256 grey levels into 2 bits (black and white) in case of grey images. The technique is even more effective in color images as it quantizes 256^3 = 16.7 color pixels into 8 color pixel. It has binary in each channel. The printing cost reduces to a very huge extent.

2. APPROACH
There are many color schemes, in terms of display we use RGB to get good visuality. But for Color half toning process we would use CMY color space. It stands for Cyan, Magenta and Yellow. CMY is subtractive color space while RGB is additive color space.
CMY color space is easy to work with, and hence half toning with error diffusion is performed on it.
The interconversion between RGB and CMY is easy and it is given by following equation:

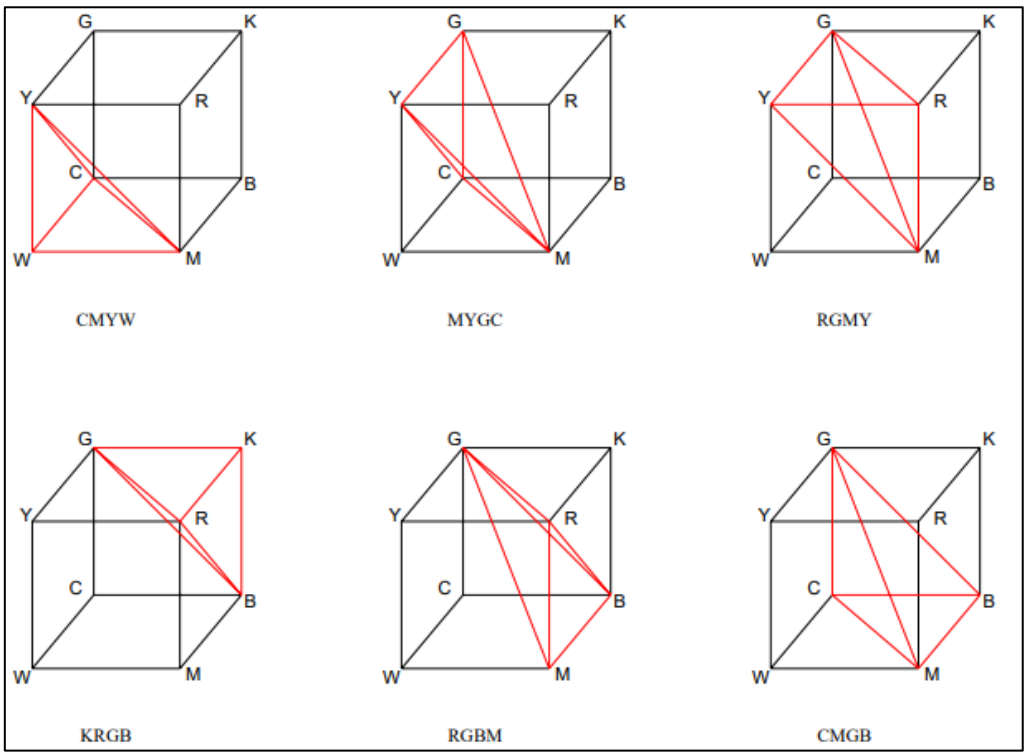
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

We use 2 approaches:
a. Separable Error Diffusion
b. MBVQ based error diffusion

Once we are in CMY color space, we perform the same tasks as we did in grey scale half toning with error diffusion method. The operation is performed each on C, M and Y pixels independently.
We use Floyd- Steinberg algorithm in error diffusion to get the result.

In MBVQ, RGB image is separated into minimum brightness variation quadrants (MBVQs). It renders RGB into one of six complementary quadruples: RGBK, WCMY, MYGC, RGMY, RGBM or CMGB. Each has minimum brightness variation. After breaking into quadruples, following steps are followed:

a. Limit pixel to one quadrant, determine MBVQ (RGB (i,j))
b. Find vertex (v) which is closet to RGB (i,j) + e(i,j)
c. Computing quantization error
d. Distribute error to next pixels.



3. RESULTS

Fig. 24 Original Color Image

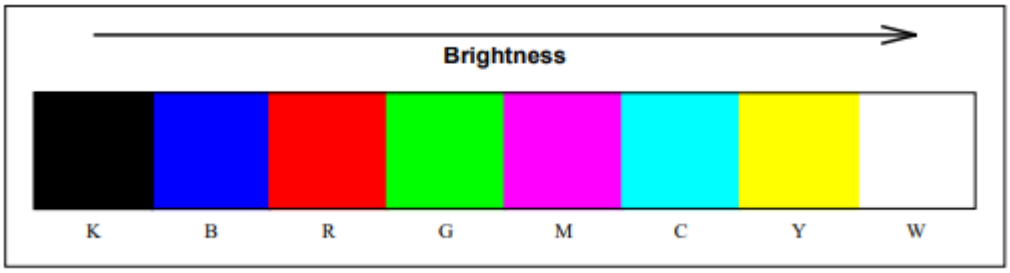
Fig. 25. Color Image after Half toning by Error Diffusion

Fig. 26. Color Image after Half toning by MBVQ

4. DISCUSSION

ANSWERS:

1. The key ideas on which MBVQ is based are as follows:
   a. Minimal Brightness Selection Criteria



   b. Minimum Brightness Variation Quadruples
      The 6 quadruples are shown in the above figure which are meant for minimum brightness.
   c. Color Diffusion Algorithm

For each pixel (i, j) in the image do:

1. Determine MBVQ$(RGB(i, j))$.
2. Find the vertex $v \in$ MBVQ which is closest to $RGB(i, j) + e(i, j)$.
3. Compute the quantization error $RGB(i, j) + e(i, j) - v$.
4. Distribute the error to "future" pixels.

It overcomes the defects of Standard Error Diffusion method is because in later we use only one color space (CMY) while in former we have 6 to choose from according to the pixel value. Also, selection of minimum brightness overcomes the defects of standard method.

2. Comparing output images of Fig and Fig, we can say that **MBVQ is better**.

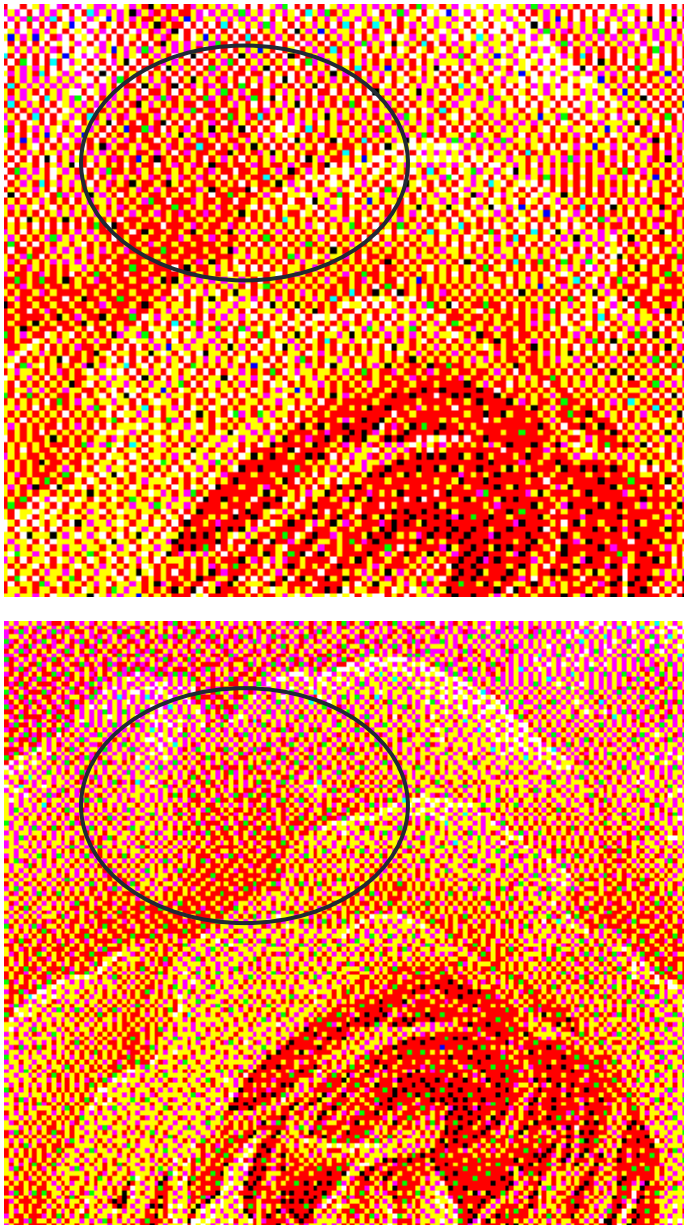


Fig. 27. Comparing Separable Error Diffusion (top) and MBVQ (bottom)

There is **decrease in halftone noise from Standard method to MBVQ.** We can see black pixels in Separable error diffusion after zooming in, in contrast to the zooming of MBVQ. In standard method, it renders the patch with all 8 halftones creeping in noise. While in MBVQ, we use only 4 colors and this reduces the halftone noise to a minimal. Artifacts in Separable error diffusion are reduced in MBVQ. MBVQ also preserves color and brightness and the results are faster.

REFERENCES:

1. https://www.researchgate.net/figure/Some-advantages-and-disadvantages-of-edge-detectors_tbl1_266287401

2. http://www.digitalxplore.org/up_proc/pdf/19-1381386785133-136.pdf

3. https://www.egr.msu.edu/classes/ece480/capstone/fall13/group04/docs/danapp.pdf

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

5. HP Labs Paper:
   https://www.hpl.hp.com/techreports/96/HPL-96-128R1.pdf

6. Comparison Floyd, JJN and Stucki:
   https://www.ijedr.org/papers/IJEDR1702301.pdf
   https://www.ijedr.org/papers/IJEDR1702306.pdf

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