EE569 Project 2

Name: Shreyas Dinesh Patil
Email: shreyasp@usc.edu

Problem 1: Edge Detection

(a) Sobel Edge detector

I. Motivation:

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. Applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified.

II. Approach:

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

Gx

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

Gy

3x3 sobel mask

Gx calculates gradient along horizontal direction and Gy along vertical direction and are numerical approximations of continuous gradients.

1) Extend the boundary of given image by 1 pixel on each side.
2) Beginning from (1, 1) as the center pixel of the gradient masks and sliding through entire extended image till (height, width) as center pixel calculate x-gradient and y-gradient.
3) Calculate the gradient magnitude for each pixel location in the input image.

4) Calculate the CDF of gradient magnitude values for the input image. Take particular percentage of CDF and use the corresponding gradient magnitude value as the threshold for generating binary edge map.
5) Normalize x and y gradients using min-max normalization to get gradient maps.

III. Experimental Results:

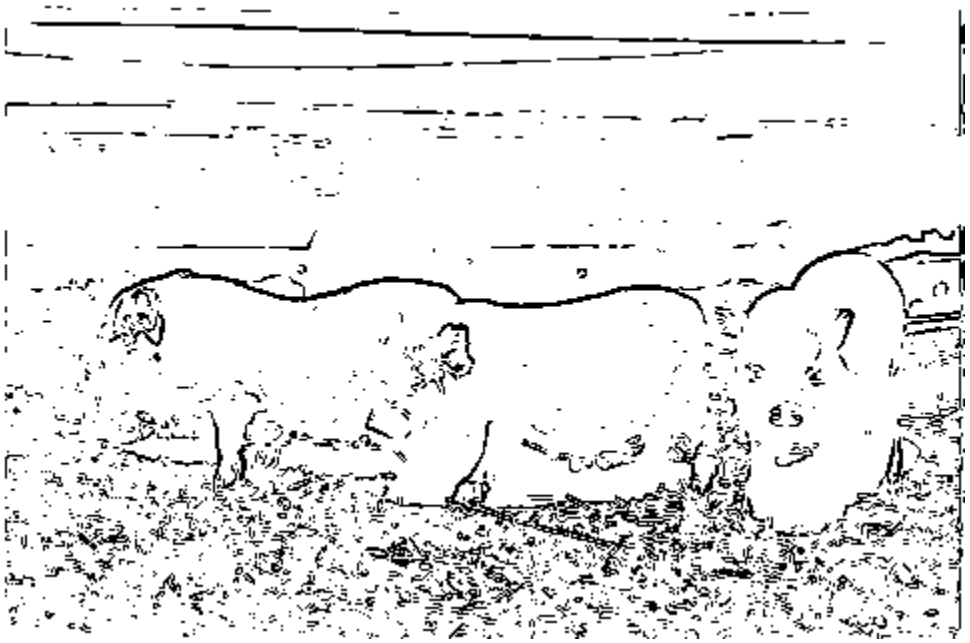


Figure1: Edge detection on Pig image with threshold=0.7

Figure2: Edge detection on Pig image with threshold=0.9

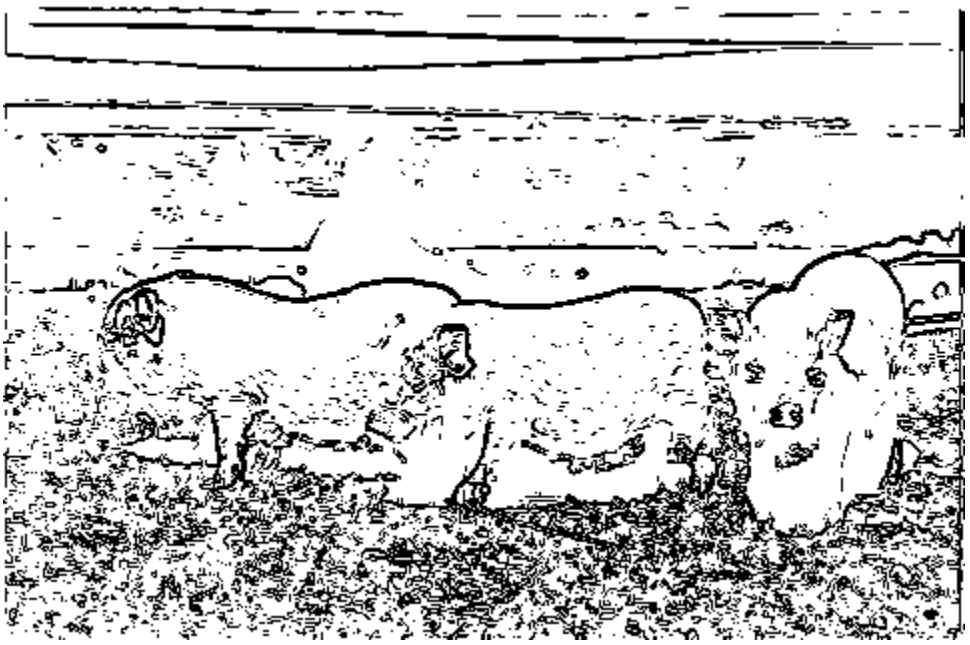


Figure3: Edge detection on Pig image with threshold=0.5

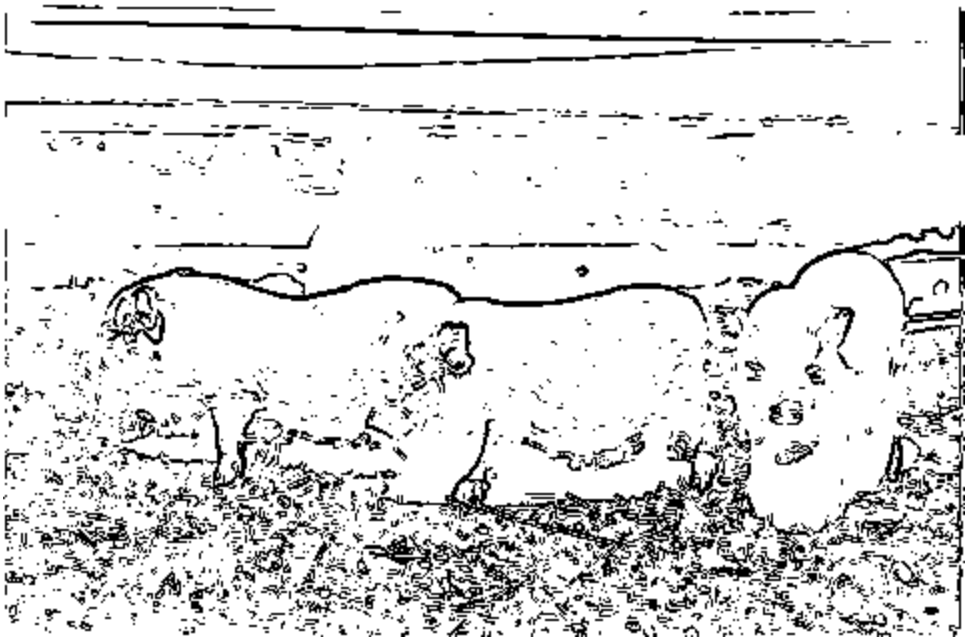


Figure4: Edge detection on Pig image with threshold=0.6

Figure5: Edge detection on Tiger image with threshold=0.6

Figure6: Edge detection on Tiger image with threshold=0.9

Figure7: Edge detection on Tiger image with threshold=0.7

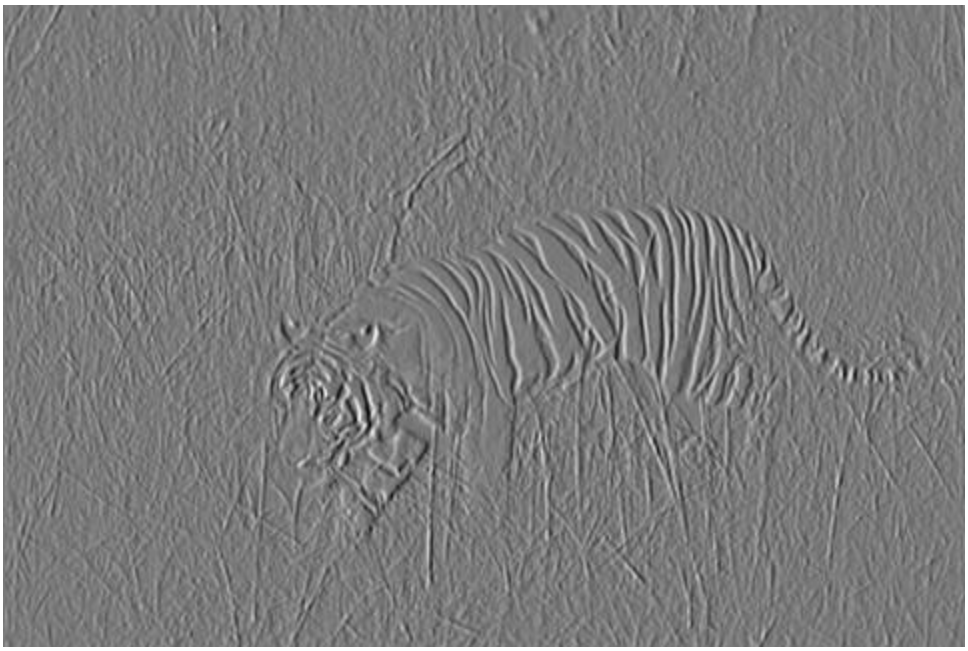


Figure8: x-gradient map (vertical edge detection)

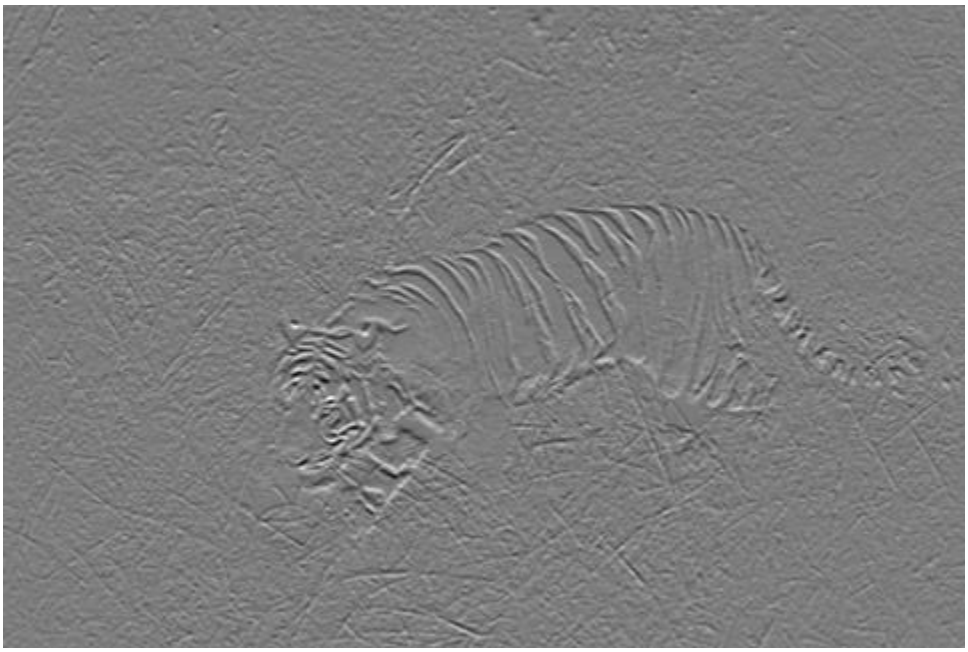


Figure9: y-gradient map (horizontal edge detection)

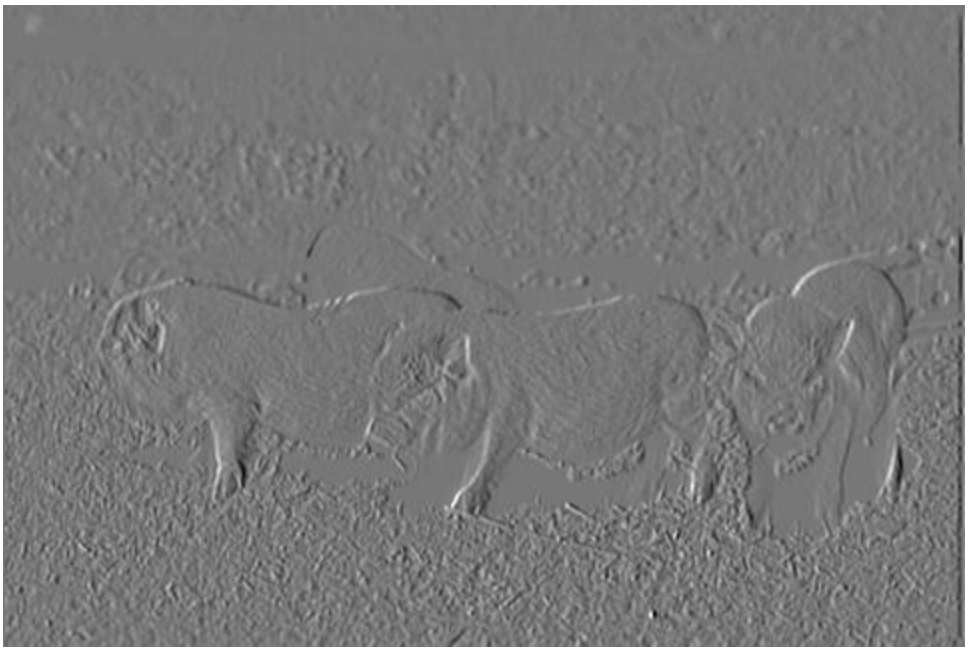


Figure10: x-gradient map (vertical edge detection)

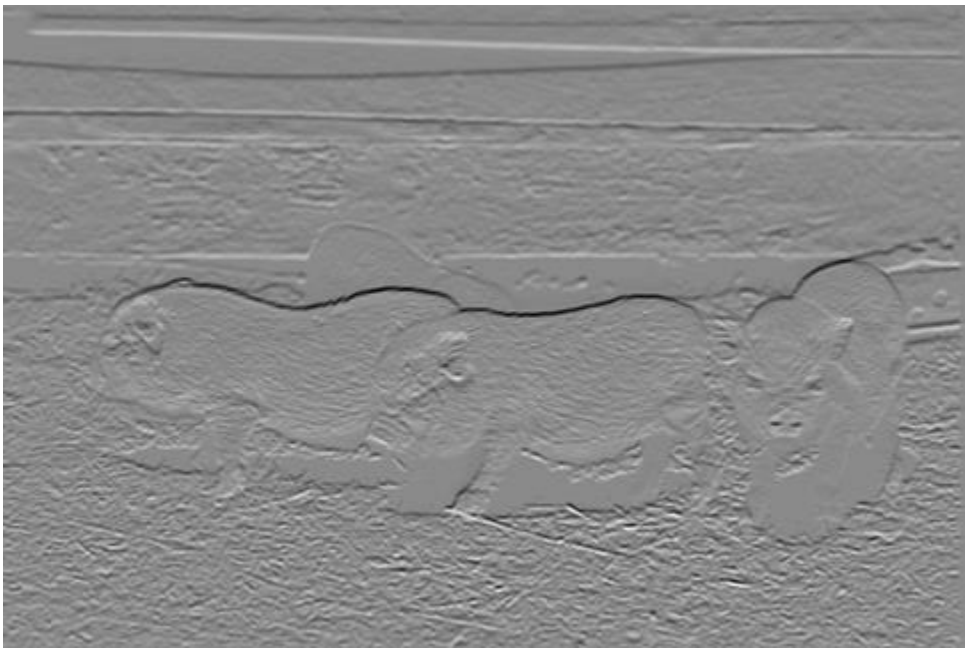


Figure11: y-gradient map (horizontal edge detection)

IV. Discussion

- 0.7percent of CDF threshold is the best edge map for pig and tiger image. As threshold decreases the sobel filter captures detail textures also in image. High threshold loses the minute details of edges and does not detect weak edges.
- The gradient maps capture the horizontal and vertical edges to a good extent.

b) Canny edge detector

I. Approach:

1) Filter out any noise in the input image using Gaussian filter.
2) Find the gradient in both x and y directions using any gradient mask (e.g. 3x3 sobel mask).
3) Find the magnitude and orientation of the gradient

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

4) Non-maximum suppression is applied. Here we suppress all gradient values to 0 except the local maxima.
5) Double thresholding is used to connect edges.

II. Experimental results:

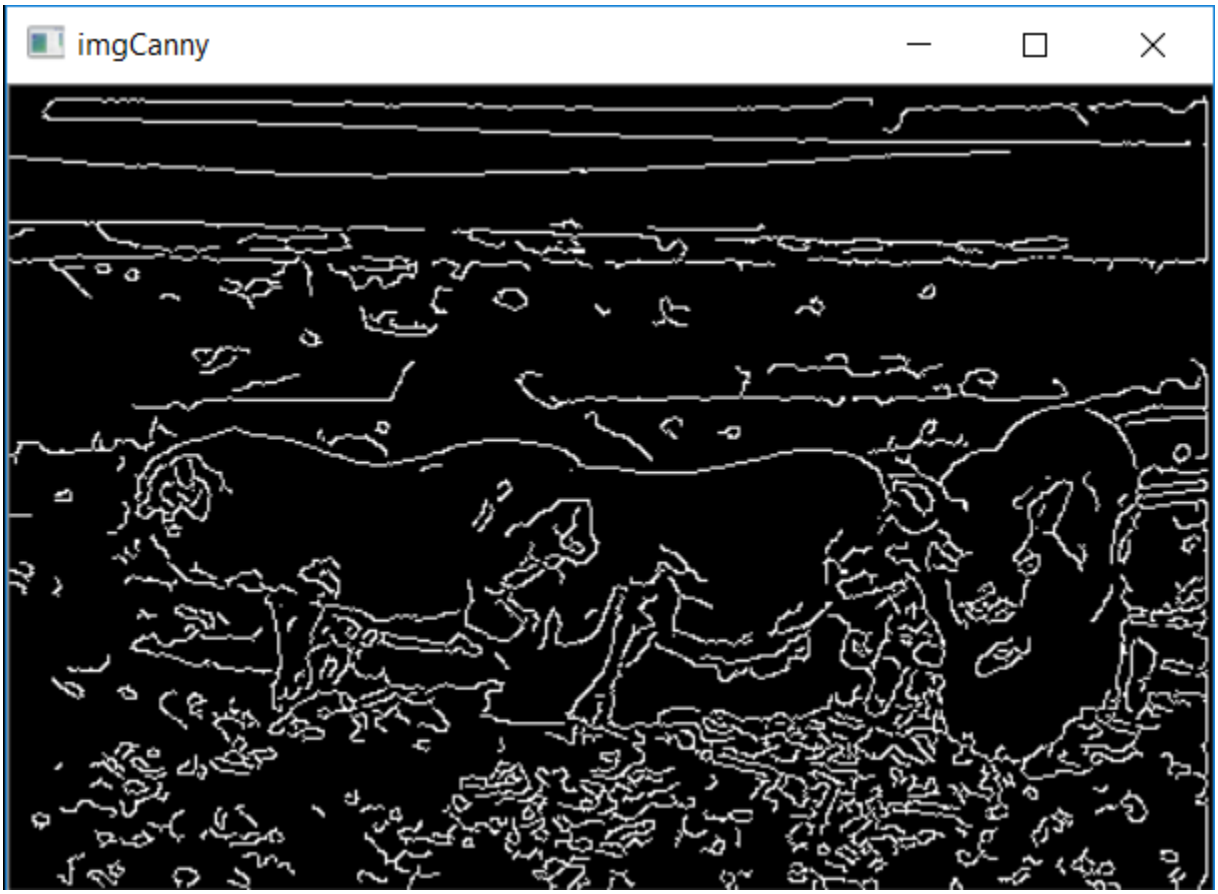


Figure1: Pig image edge detection with upper (100) and lower (50) threshold value ratio (2:1)

Figure2: Pig image edge detection with upper (150) and lower (50) threshold value ratio (3:1)

Figure3: Pig image edge detection with upper(125) and lower(50) threshold value ratio(2.5:1)

Figure4: Tiger image edge detection with upper(150) and lower(50) threshold value ratio (3:1)

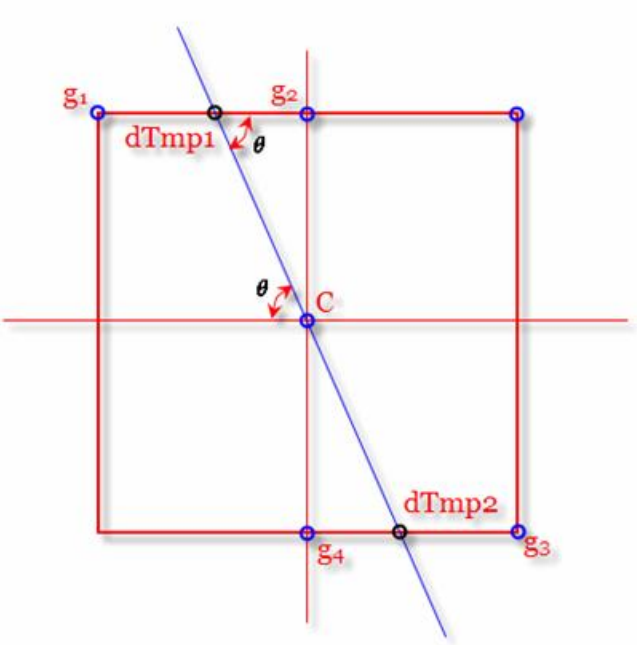
Figure5: Tiger image edge detection with upper(100) and lower(50) threshold value ratio (2:1)

Figure6: Tiger image edge detection with upper and lower threshold value ratio (2.5:1)

III. Discussion:

- From the above figures we can observe the effect of changing threshold value ratio between upper and lower threshold. Ratio of 2:1 captures textures also along with edges. Ratio 3:1 has less textures compared to 2:1.
- As the ratio between upper and lower threshold increases less edges are detected. If ratio decreases edges are detected along with textures. From the above figures we can see ratio 2:1 gives good results.

b) 1. Non-maximum suppression:



1) Non-maximum suppression removes pixels that do not belong to edge. That is it makes edges thin.
2) In this procedure we compare the gradient magnitude of the current pixel and compare it with gradient magnitude of pixels in negative and positive directions.
3) If the gradient magnitude is highest compared to other pixels in the mask in the same direction (the pixel that is pointing in the y-direction, it will be compared to the pixel above and below it in the vertical axis) the value will be preserved else suppressed.

b) 2.

1) If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge
2) If a pixel gradient value is below the lower threshold, then it is rejected.

If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

b) Structured Edge Detection

I. Approach

1) For training we are given a set of segmented training images. Each image has a label which consists of hand annotated edges in the image.
2) From each 32x32 patch in the input image 7228 total candidate features per patch are generated for each of the 3 RGB channels.
3) Map the structured space to output space for mapping output segmentation masks to discrete classes.
4) Train the Random forest classifier with created training data.
5) Extract features same as extracted for training data and use trained structured random forest classifier for prediction of edges in test image data.

II.Experimental results

Figure1: Original tiger image

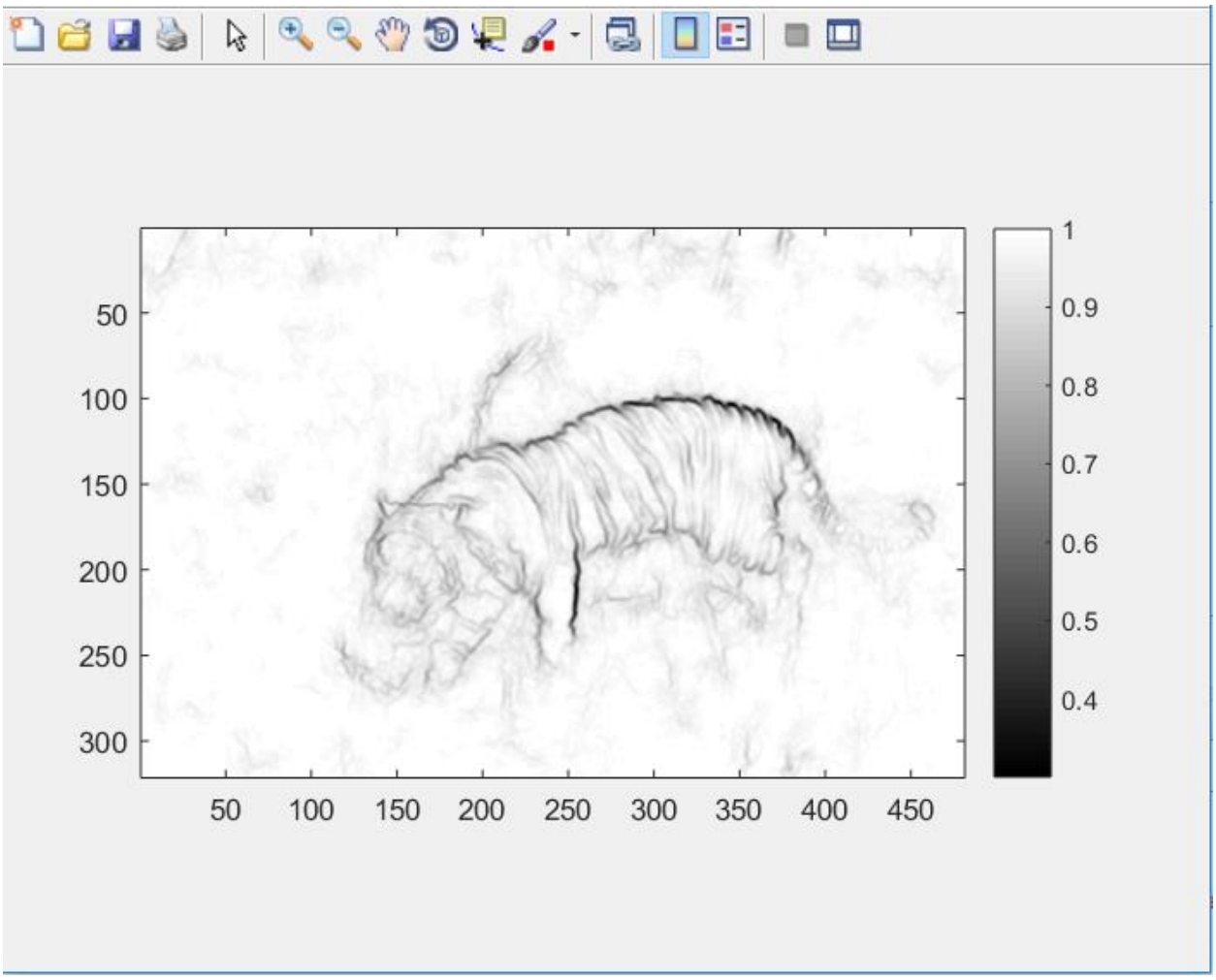


Figure2: Probability edge map for tiger image

Figure3: Binary map for tiger image using threshold 0.85

Figure4: Binary map for tiger image using threshold 0.95

Figure5: Binary map for tiger image using threshold 0.5

Figure6: Original pig image

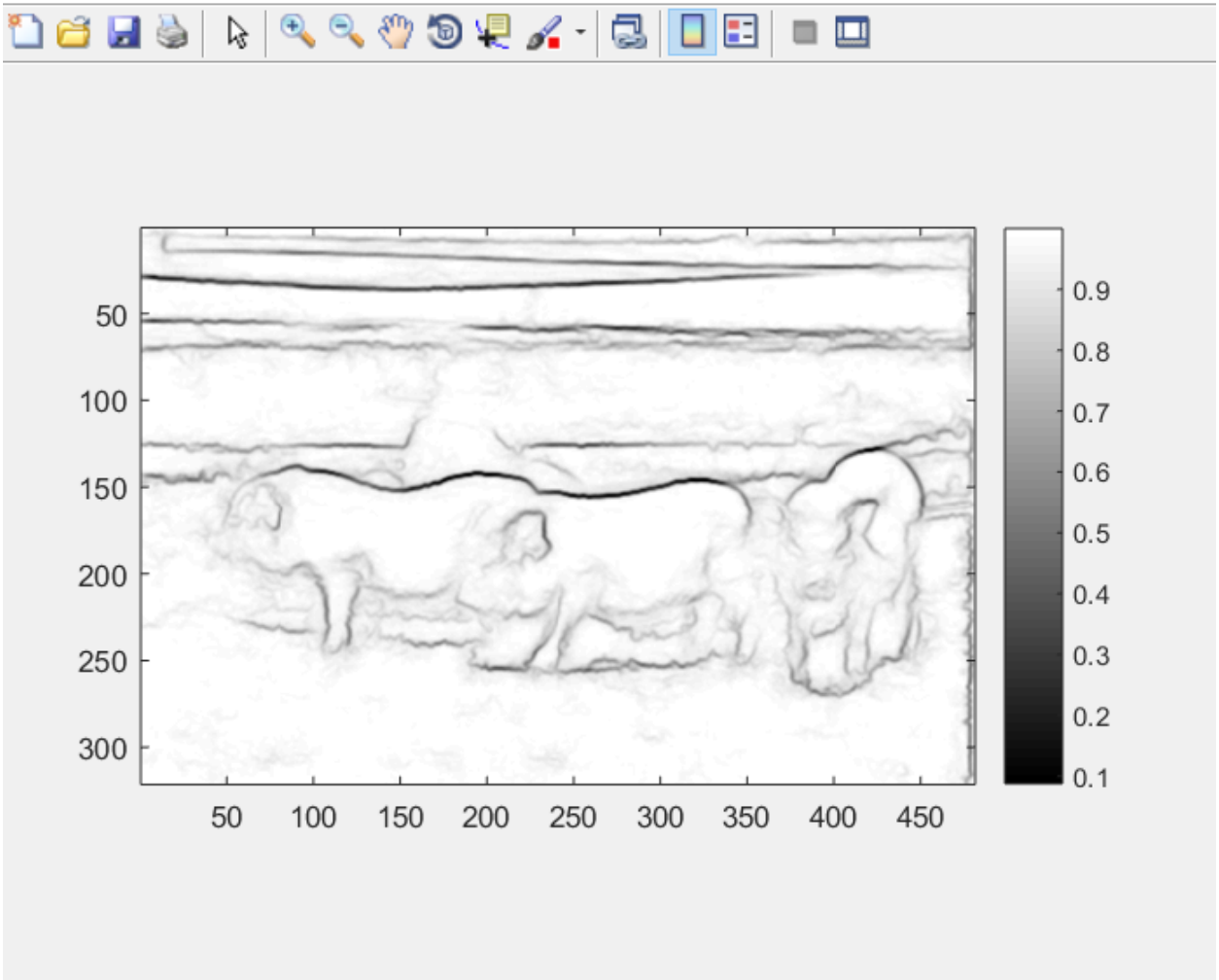


Figure7: Probability edge map for Pig image

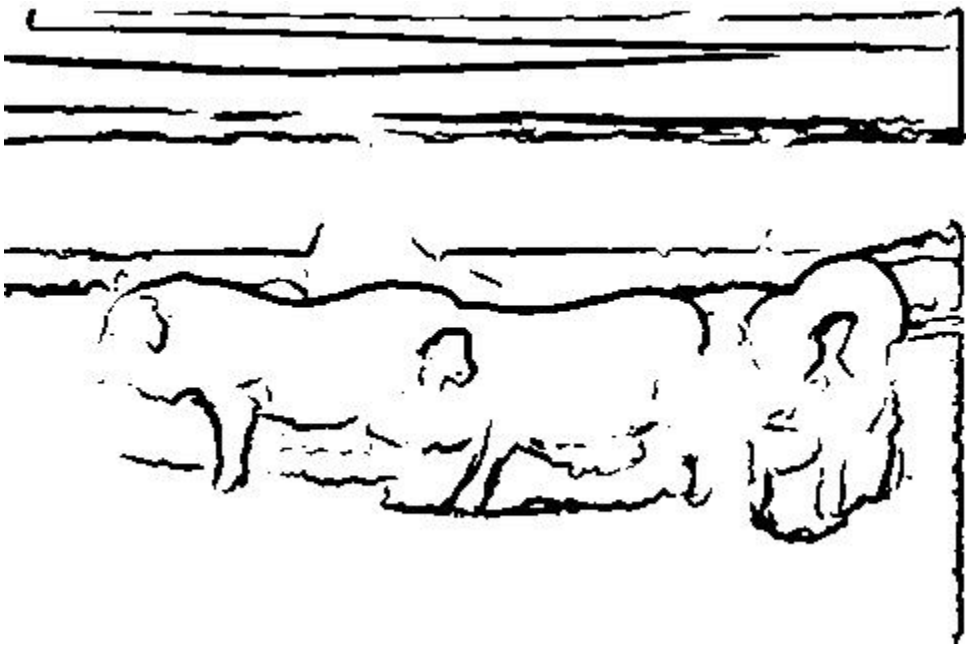


Figure8: Binary map for pig image using threshold 0.85

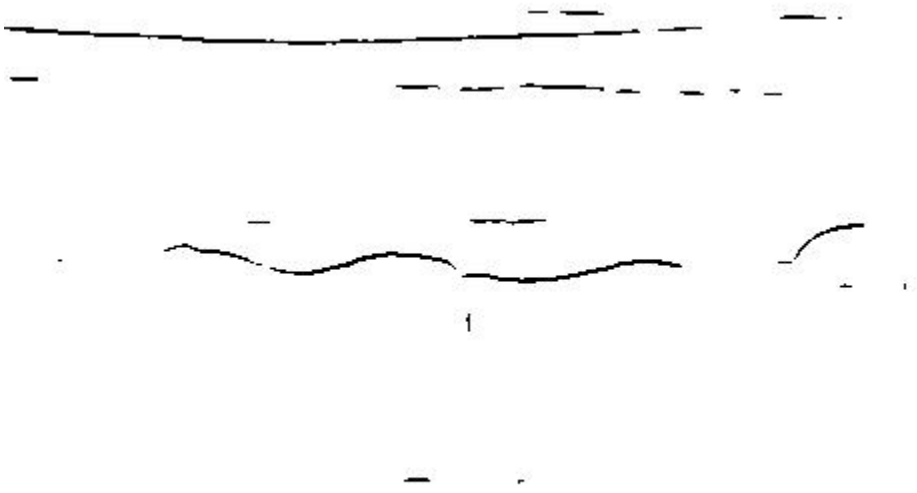
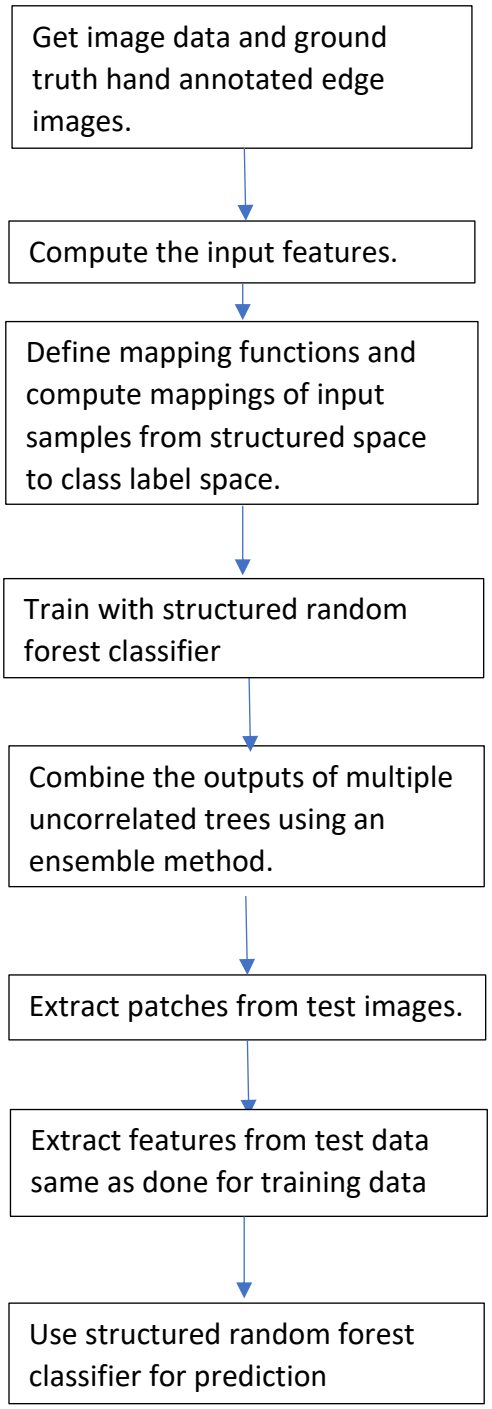


Figure9: Binary map for pig image using threshold 0.5

Figure10: Binary map for pig image using threshold 0.95

III.Discussion:



1) For training we are given a set of segmented training images. Each image has a label which consists of hand annotated edges in the image.
2) From each 32x32 patch in the input image 7228 total candidate features per patch are generated for each of the 3 RGB channels.
3) Map the structured space to output space for mapping output segmentation masks to discrete classes.
4) Train the Random forest classifier with created training data.
5) Extract features same as extracted for training data and use trained structured random forest classifier for prediction of edges in test image data.

1. A decision tree classifies instance $x \in X$ by recursively branching left or right down the tree until a leaf node is reached. Each node j in the tree is associated with a binary split function. The feature to split is decided by a score function. The feature which maximizes the score function is chosen for split.
   The stopping criteria for splitting of nodes can be any of the following:
   1. Until all leaf nodes are reached
   2. Until particular depth of the tree
   3. Maximum number of leaf nodes allowed.

   Each leaf node represents a class.
   A decision forest is an ensemble of T independent trees. Given a sample x, the predictions ft(x) from the set of trees are combined using an ensemble model into a single output.
   Decision trees are highly prone to overfitting therefore the data is randomly sampled into subset samples which are given to individual decision trees. This makes the trees independent of each other.
   The final output of random forest is average of outputs of decision trees for regression and majority class obtained as output of decision trees in classification tasks.

2. The threshold is 0.85 because it gives the best edge map compared to other threshold values. Low threshold values cannot detect sufficient edges and high threshold values detect textures and also thicken the edges. Canny has thin edges while structured edge has thick edges. Canny has more extracted textures which are relatively less than structured edge. Both have continuous edges.

d) Performance Evaluation

I. Experimental results

For Sobel

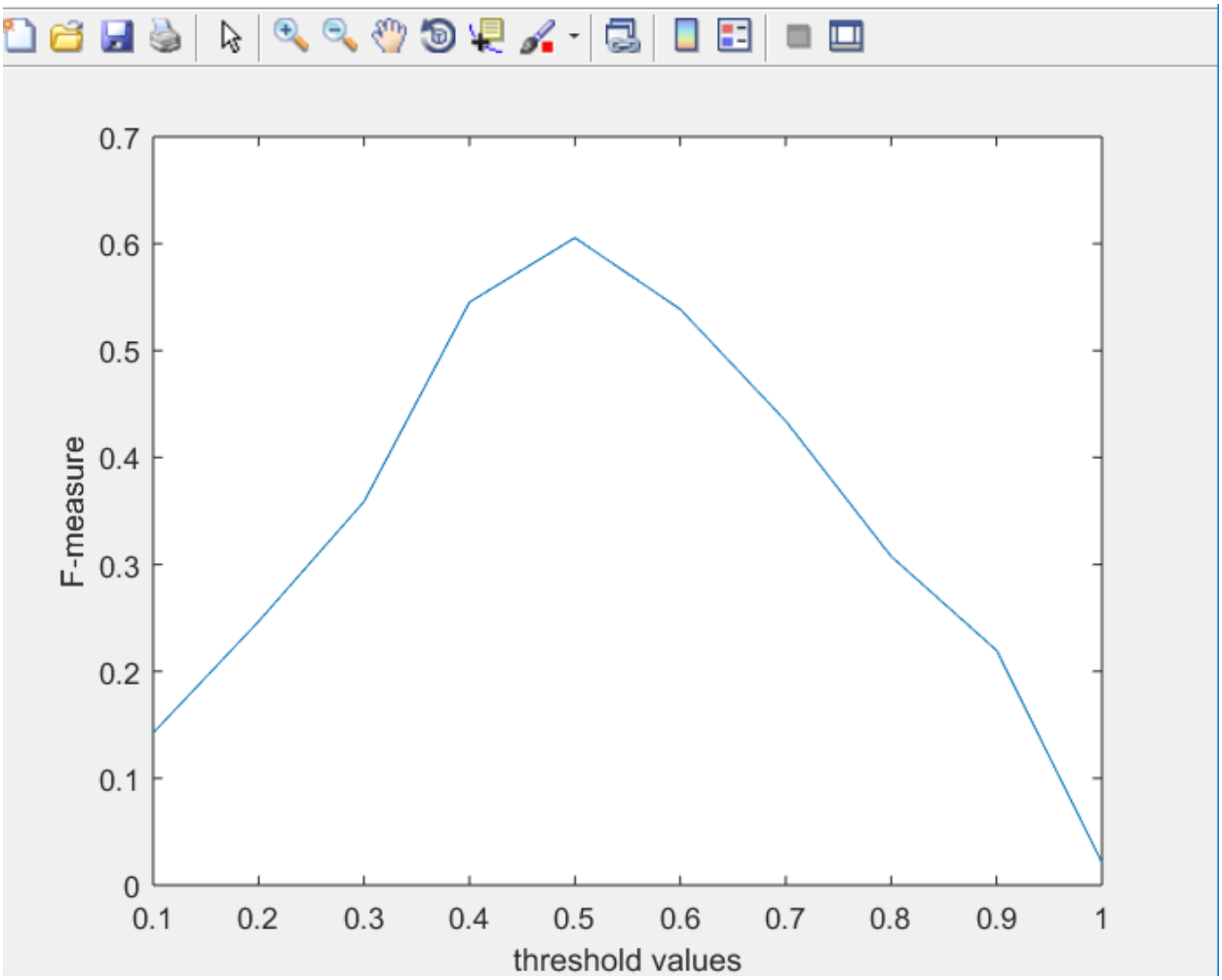


Figure1: For probability Tiger image

Best_F_measure = 0.6059

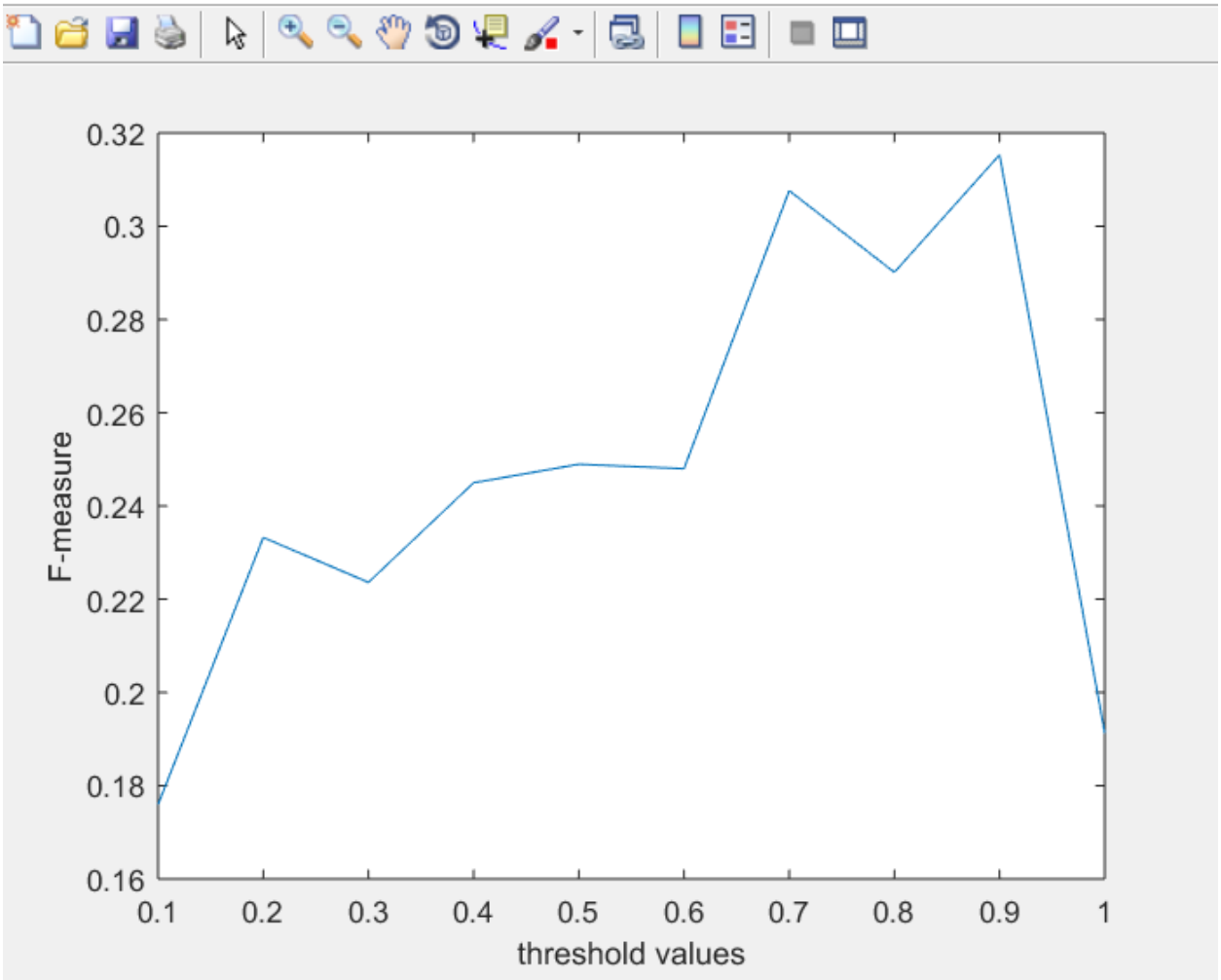


Figure2: For Pig probability image

Best_F_measure = 0.3149

For Canny

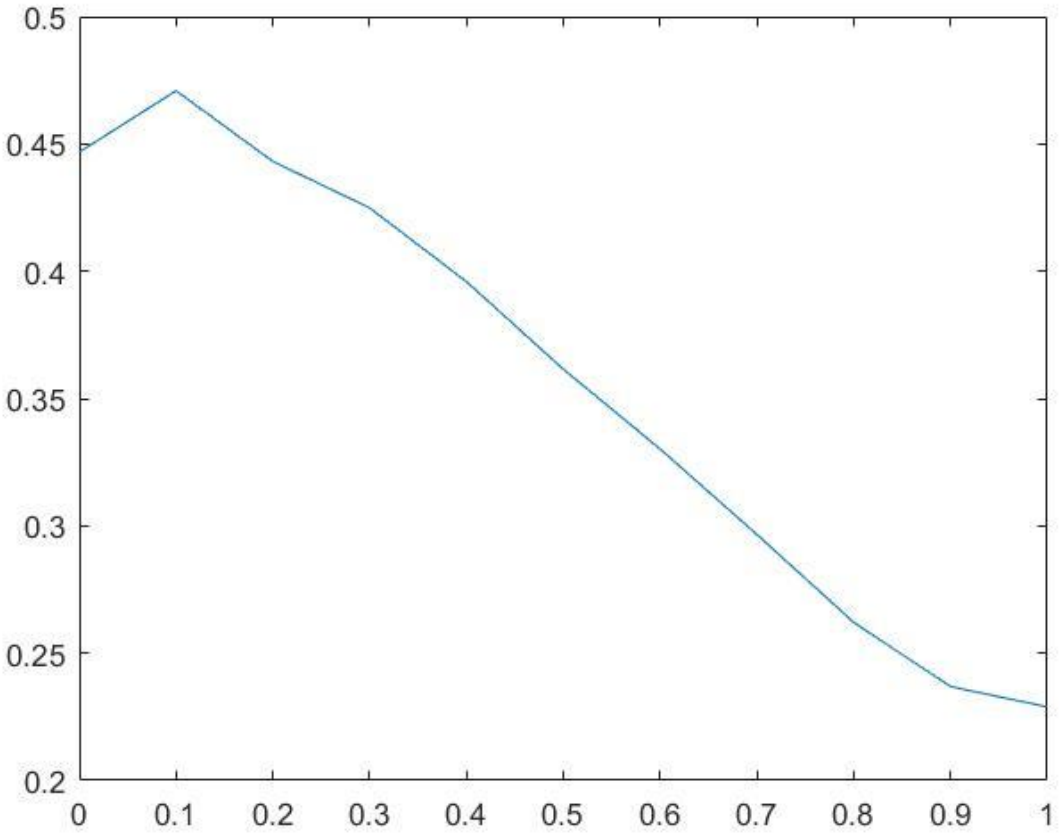


Figure3: For Tiger probability image

Best_F_measure = 0.4010

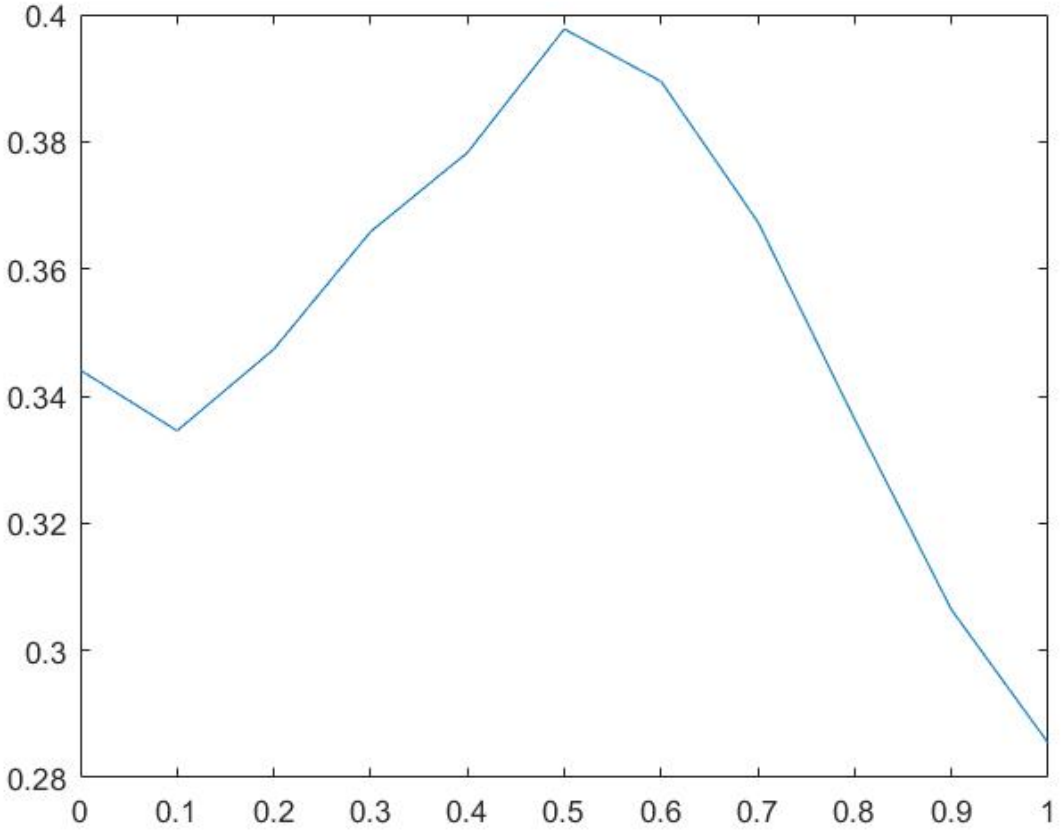


Figure4: For Pig probability image

Best_F_measure = 0.3969

For Structured Edge

For Pig

Ground Truth	Precision	Recall	Mean Precision over thresholds	Mean Recall over thresholds	Final F measure
G1	0.1328	0.2382	0.1120	0.2245	0.1494
G2	0.1509	0.2503	0.1193	0.2251	0.1560
G3	0.2331	0.24148	0.2016	0.2543	0.2249
G4	0.3056	0.2518	0.2388	0.2441	0.2414
G5	0.2665	0.2828	0.1974	0.2436	0.21808

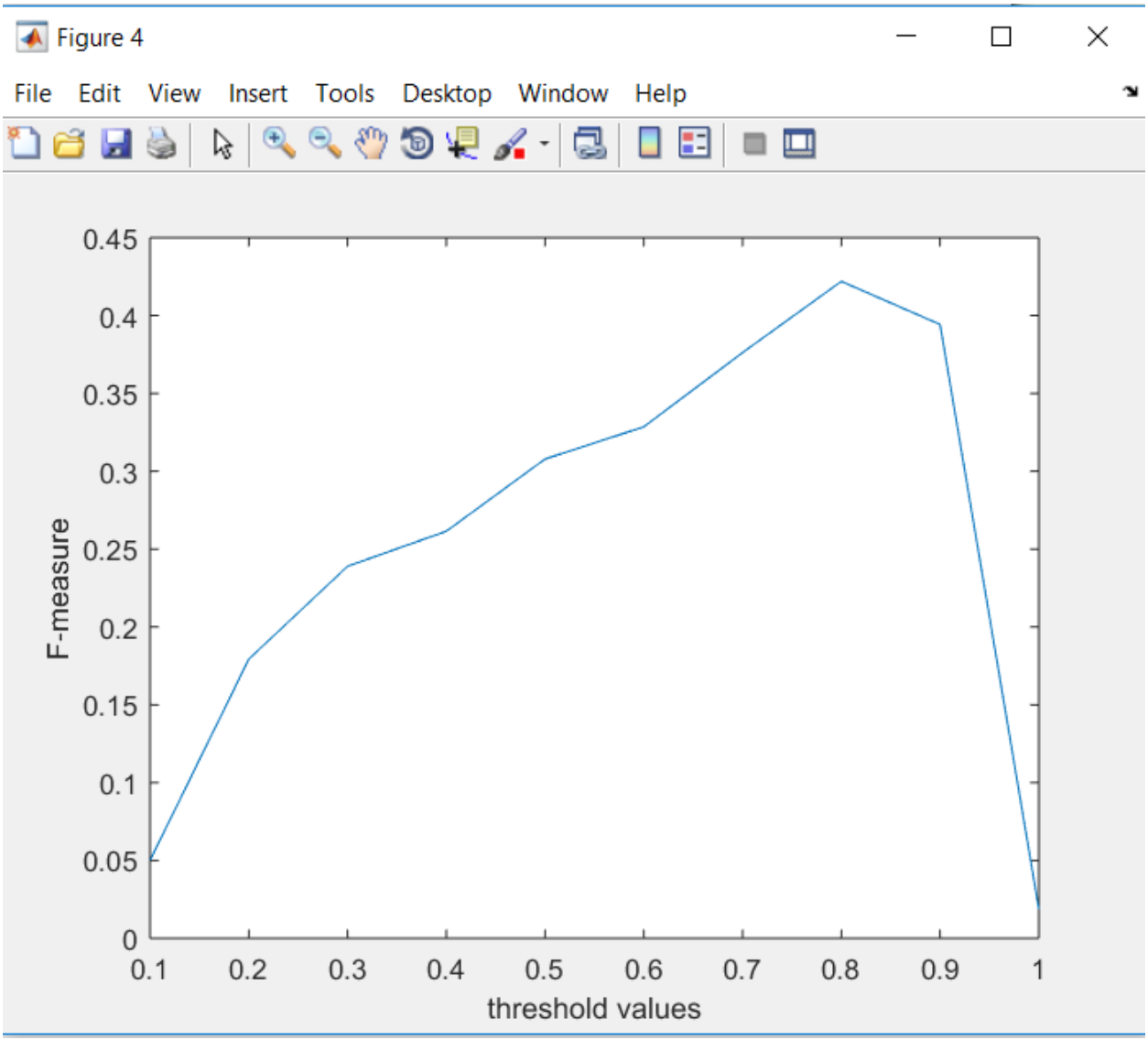


Figure5: For Pig probability image

For Tiger

Ground Truth	Precision	Recall	Mean Precision over thresholds	Mean Recall over thresholds	Final F measure e
G1	0.1258	0.2804	0.0625	0.1716	0.0916
G2	0.1291	0.2713	0.0659	0.1738	0.0956
G3	0.1517	0.2840	0.0740	0.1721	0.1035
G4	0.5518	0.2620	0.5679	0.1795	0.2728
G5	0.1278	0.2063	0.1252	0.1545	0.1384

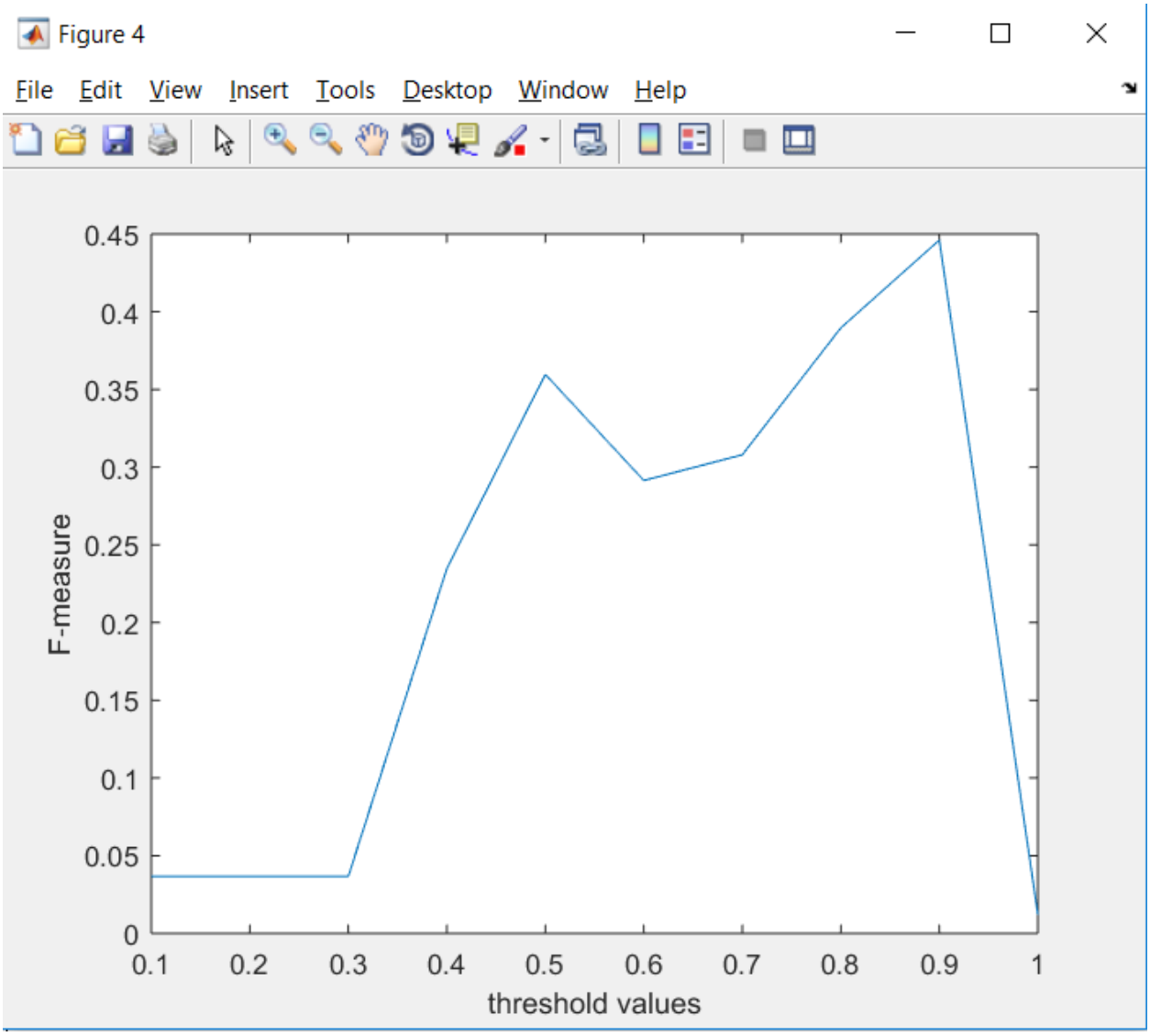


Figure6: For Tiger probability image

II. Discussion

1) Canny edge detector gives thinner and continuous edges. Needs manual tuning of thresholding.
   It is very simple to implement. Sobel gives good results in presence of low noise. It can detect edges and their orientations Sobel fails to detect weak edges. Sobel filter cannot produce accurate edge detection with thin and smooth edge.

2) Pig image is easier to get a higher F-measure for same threshold value. Pig image contains less textures compared to tiger image, therefore the balance between precision and recall is achievable for pig image.
3) We take harmonic mean of Precision and recall because it punishes extreme values. If we have precision 1 and recall 0 we get harmonic mean as 0. Thus, it punishes extreme values.

   High F-measure means high precision and recall.

   No, it is not possible to get high F measure if precision is low and recall is high and vice versa.

   F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives). It gives equal weightage to precision and recall.

   Derivation:

   P+R = C (constant);

   F = 2*P*R/(P+R);

   F = 2*(R-C)*R/C

   Taking derivative w.r.t. R, we get 4*R = 2*C

   Which implies, P=R

   Therefore F-measure reaches maximum value when precision equals recall.

Problem 2: Digital Halftoning

I. Motivation:

Since most printing devices are not able to reproduce different shadows of gray or color images the original digital image has to be transformed into an image containing white (0's) and black (1's) for grayscale images and 8 colors of RGB model for color images.

Halftoning is important since it is used by printing machines.

a) Dithering

1) Random Thresholding:

II. Approach:

1) For each pixel, generate a random number in the range 0 ~ 255, so called $rand\ (i, j)$
2) Compare the pixel value with $rand\ (i, j)$. If it is greater, then map it to 255; otherwise, map it to 0,

G (i, j) = 0 if 0 <= F (i, j) < rand (i, j)
255 If rand (i, j) <= F (i, j) < 256

Here random threshold is uniformly distributed random variable.

2) Dithering Matrix

II. Approach:

- Bayer's initial index matrix is given by,

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

where 0 indicates the pixel most likely to be turned on, and 3 is the least likely one. Family of dithering Bayer's matrices can be computed recursively using,

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

The index matrix can then be transformed into a threshold matrix T for an input gray-level image with normalized pixel values (i.e. with its dynamic range between 0 and 255) by the following formula:

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

N^2 – no. of pixels in the matrix

Halftoned image pixel values are calculated based on the following condition,

$G\ (i, j) = 0\ if\ 0 \le F\ (i, j) \le T(i\ modN, j\ modN)$

$255 \qquad T(i\ modN, j\ modN) < F\ (i, j) < 256$

III. Experimental results

Figure1: Halftone result on applying 2x2 dithering matrix

Figure2: Halftone result on applying 8x8 dithering matrix

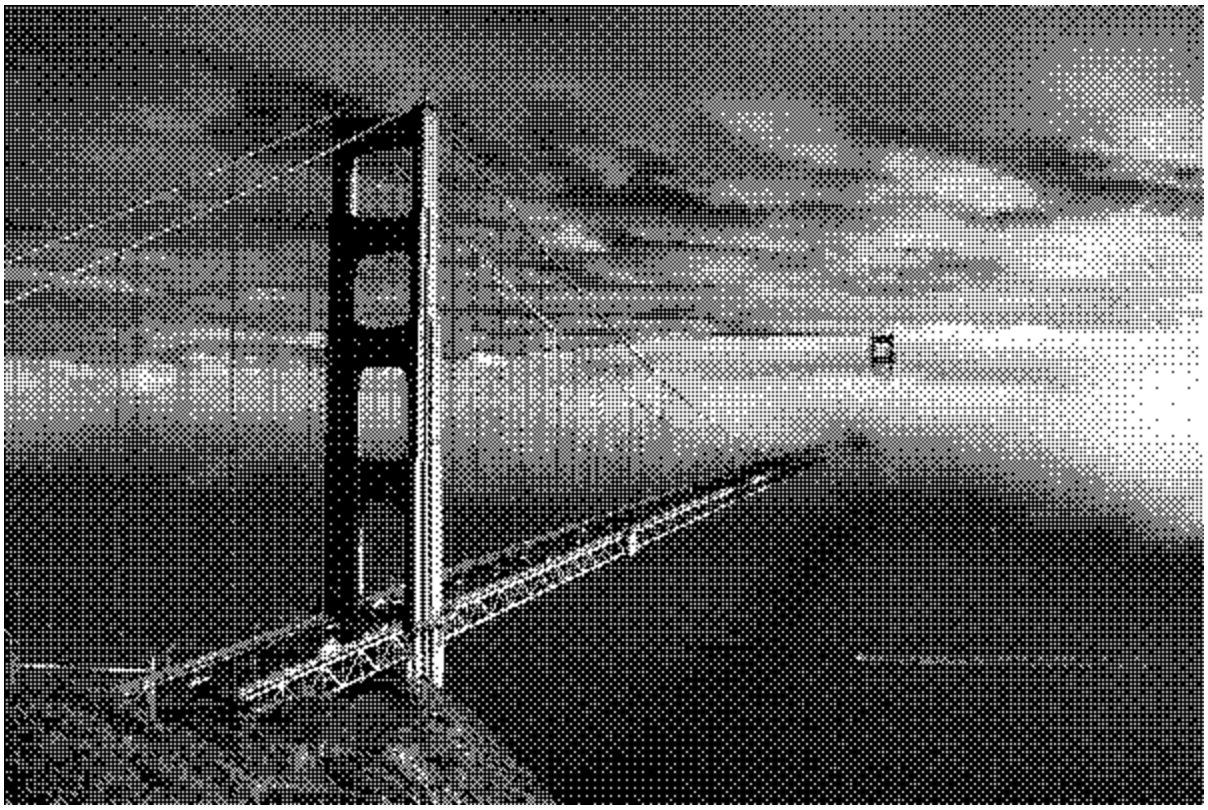


Figure3: Halftone result on applying 32x32 dithering matrix

IV Discussion

- The results obtained using 2x2 dithering matrix are the worst. In this image there is not gradual change of tone.
- The results obtained for 8x8 and 32x32 dithering matrix are better. But the results contain box-like pattern repeating this is due to the structure of threshold matrix. The results obtained using these dithering matrices are better than 2x2 and are visually near to the original.

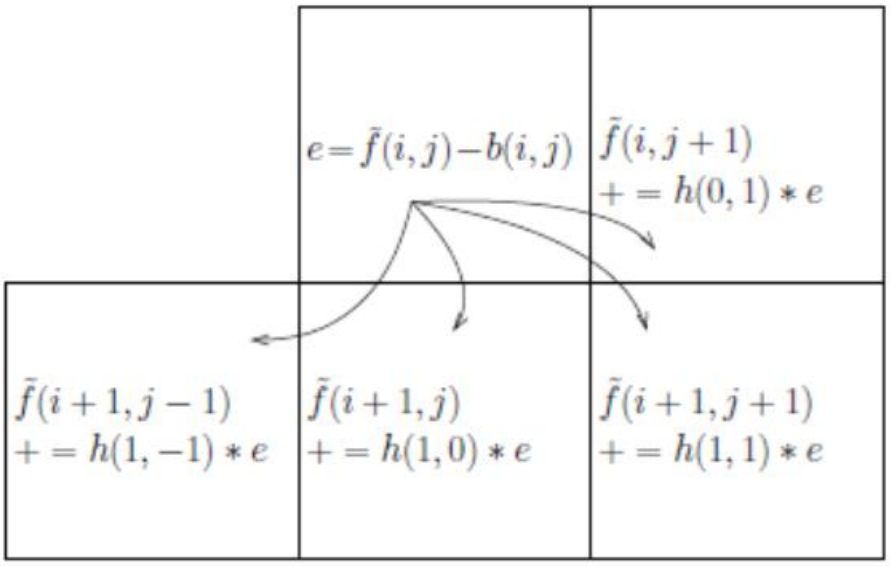
b. Error Diffusion

I. Motivation

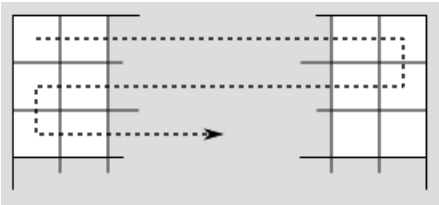
In error diffusion we compute the error caused by thresholding a given pixel and propagate it to neighbor pixels, to compensate for the average intensity loss or gain. It is based upon the conception that a slightly out-of-place pixel causes little visual harm.

II. Approach

- Error diffusion is based on the simple principle that once a pixel has been quantized, thus introducing some error, this error should affect the quantization of the neighboring pixels. The way the error is affecting the quantization of its neighboring pixels is referred to as diffusion, meaning that the error is split in a few components and then added to the gray level values of the neighbors. By diffusing the error, the system performs as a self-correcting, negative feedback system.
- Extend the image on all sides by 1 pixel if error diffusion matrix is 3x3 and by 2 pixels on all sides if it is 5x5.
- Each pixel value is quantized using fixed threshold value.
- The quantization error of center pixel is then diffused to neighboring pixels falling in error diffusion filter weighted by the filter matrix coefficients. The below diagram illustrates that.



- The diffusion matrix is slided over the entire image in serpentine manner as shown,



While going from left to right in the image the error diffusion matrix is flipped.

c. Experimental Results

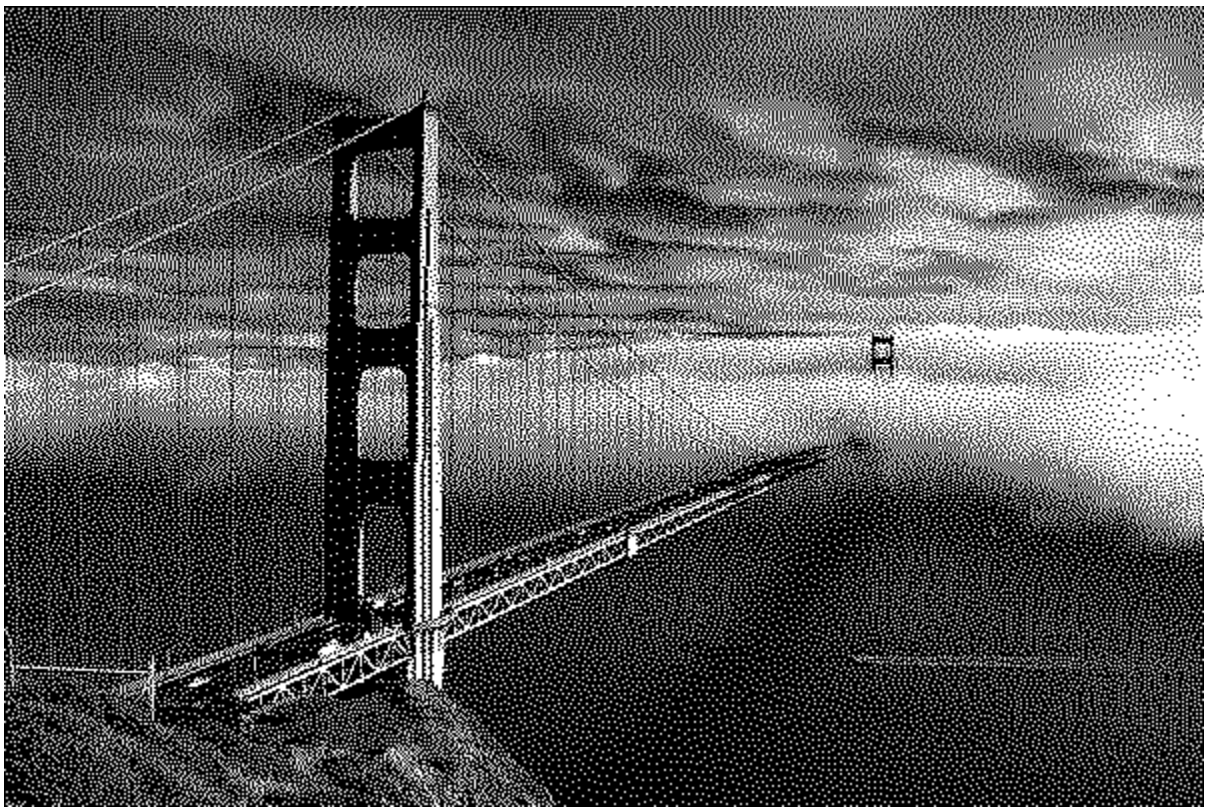


Figure1: Halftone result on applying Floyd Steinberg error diffusion matrix

Figure2: Halftone result on applying JJN error diffusion matrix

Figure3: Halftone result on applying Stucki error diffusion matrix

IV. Discussion

- Floyd-Steinberg dithering only diffuses the error to neighboring pixels. This results in very fine-grained dithering.
- JJN dithering diffuses the error also to pixels one step further away. The dithering is coarser but has fewer visual artifacts. However, it is slower than Floyd-Steinberg dithering because it distributes errors among 12 nearby pixels instead of 4 nearby pixels.
- Stucki is slightly faster. Its output tends to be clean and sharp.
- From above figures we observe that error diffusion method gives better half toned images. There are no block patterns repeating in error diffusion half-toning.
- I prefer error diffusion method. It looks visually more near the original image and it does not have visual artifacts. By diffusing the error, the system performs as a self-correcting, negative feedback system reducing patterns in the output.
- Multiscale error diffusion can achieve better results. In this method the order in which is random in the sense that it is image dependent. In this method image pixels are scanned in a way determined by their local intensity. Upon quantization of an image pixel, the diffusion of the quantization error prohibits the accumulation of the error locally. These features ensure the there is gradual change of tone in the image.

b) Color Halftoning

I. Approach

- Convert RGB image into CMY using
- Extend boundary of CMY image by 1 on each side (if using Floyd-Steinberg error diffusion).
- Apply the Floyd-Steinberg error diffusion algorithm to quantize each channel separately.
- After applying Floyd-Steinberg error diffusion convert back the image to RGB.

II. Experimental results

Figure1: Original bird image

Figure2: Color Halftone result on applying separable error diffusion method

III.Discussion

- As you can see the color toned image has tiny granular structures. These are due to the Floyd-Steinberg error diffusion matrix.
- Monochrome halftone algorithms are designed to reduce visible artifacts. Important factor producing those artifacts is variation of brightness of the dots. In binary halftone this factor cannot be mitigated. In this method 3 halftoned monochrome planes corresponding to the image overlooks the fact that color dots are not equally bright. This is the main drawback of this approach.

I. Approach

- Extend the image boundaries by 1 pixel on each side.

1) For each pixel (i, j) in the original image do:
    1. Determine MBVQ (RGB(i, j)) using,

```
pyramid MBVQ(BYTE R, BYTE G, BYTE B)
{
if((R+G) > 255)
        if((G+B) > 255)
                if((R+G+B) > 510) return CMYW;
                else return MYGC;
        else return RGMY;
else
```

```
if (! ( (G+B) > 255) )
        if ( ! ( (R+G+B) > 255))  return KRGB;
        else return RGBM;
else return CMGB;
}
```

2. Find the vertex v belonging to 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 using Floyd-Steinberg error diffusion.

MBVQ determines maximal brightness variation quadruple in RGB cube. This reduces the number of participating color halftones to 4.

II. Experimental results

Figure1: Color Halftone result on applying MBVQ error diffusion method

III.Discussion

1) Color in RGB can be rendered using 8 basic colors located at the vertices of the cube. MBVQ renders RGB color using 4 colors and reduces halftone noise by selecting from within the halftone sets (sets of 4 colors) by which the desired color may be rendered, the one whose brightness variation will be minimal.
   By using minimal brightness variation criterion, the color patches appear more saturated. The colors used in color halftone images reduces the notice-ability of pattern which cannot be satisfied by simple error diffusion applied to 3 monochrome planes separately.

2) Visually MBVQ error diffusion and separable error diffusion have no much difference. MBVQ based algorithm requires no additional memory and entails a reasonable increase in run-time.

References

1) P. Dollár and C. L. Zitnick, “Structured forests for fast edge detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1841–1848.
2) D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones", *HP Labs Technical Report*, HPL-96-128R1, 1996.
3) A Multiscale Error Diffusion Technique for Digital Halftoning Ioannis Katsavounidis and C.-C. Jay Kuo