EE669 HW2

Problem 1: Edge Detection

a) Sobel Edge Detector

Motivation:

In broad strokes, 'edges' in images are related to gradients, which motivated their development of a discrete differentiation operator.

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

Approach:

- Convert RGB image to Grayscale.
- Find edge by checking gradient.
- To compute Gradient,
  Compute the finite difference at the center pixel by convolution

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

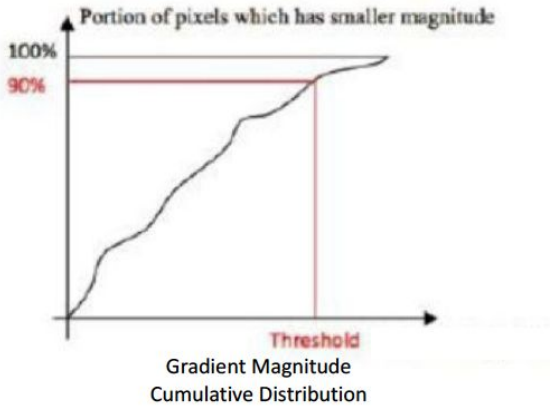
Gx

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

Gy

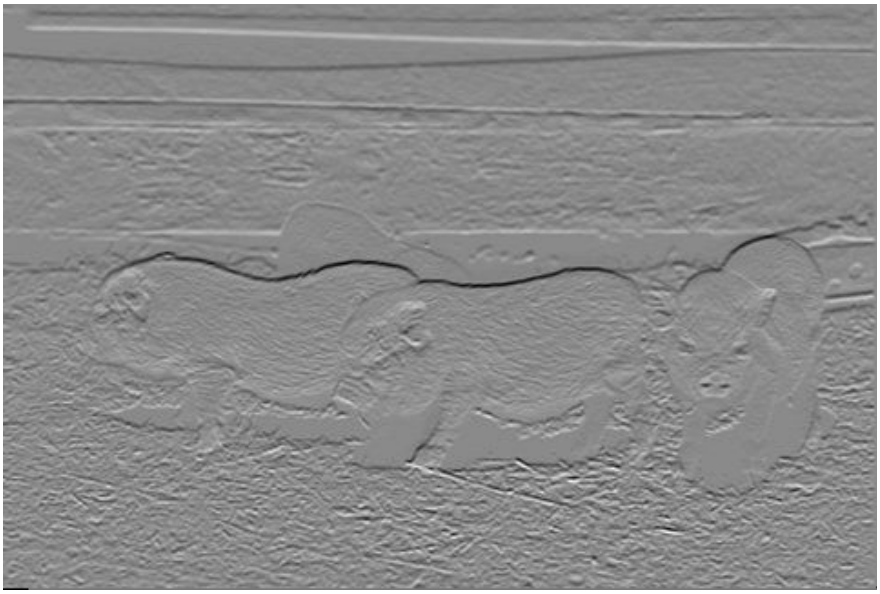
3-by-3 Sobel mask

- Normalize results for output (0~255)
- Find CDF of Gradient values and use ad-hoc method to find Threshold

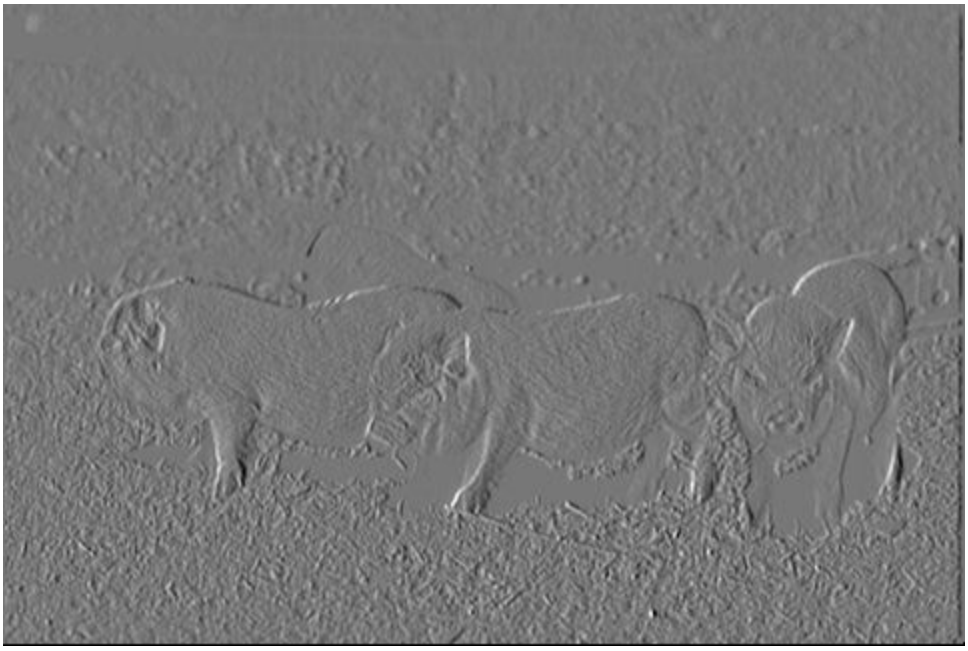


Experimental Results:

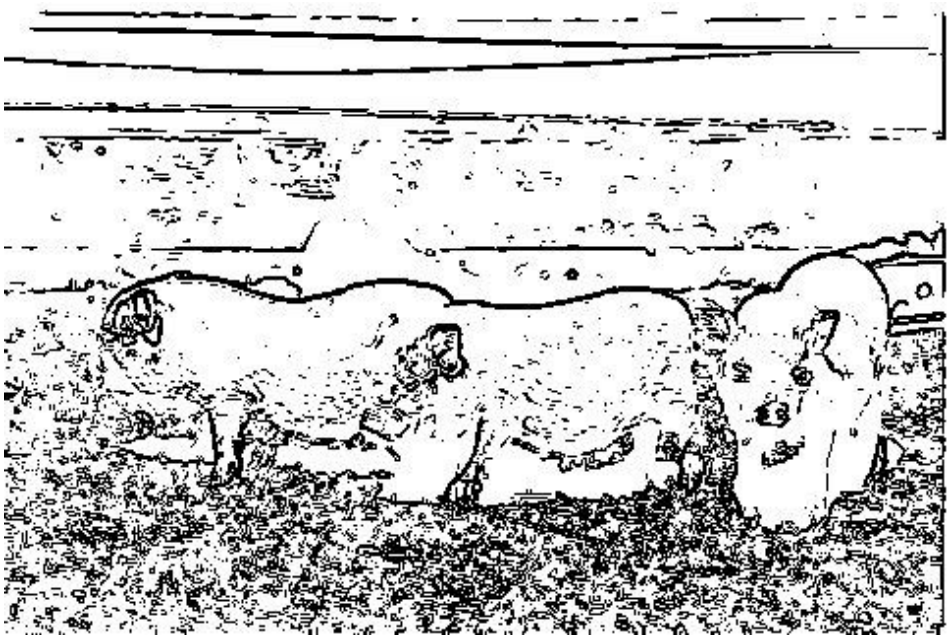
Threshold of 80%:



y_gradient Edge Map

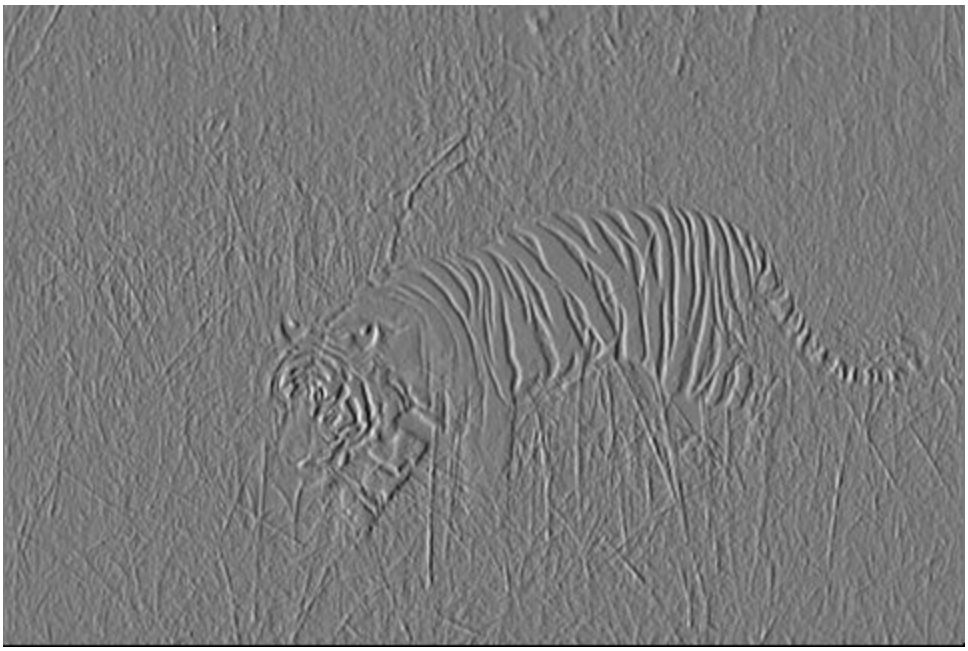


X_gradient Edge Map

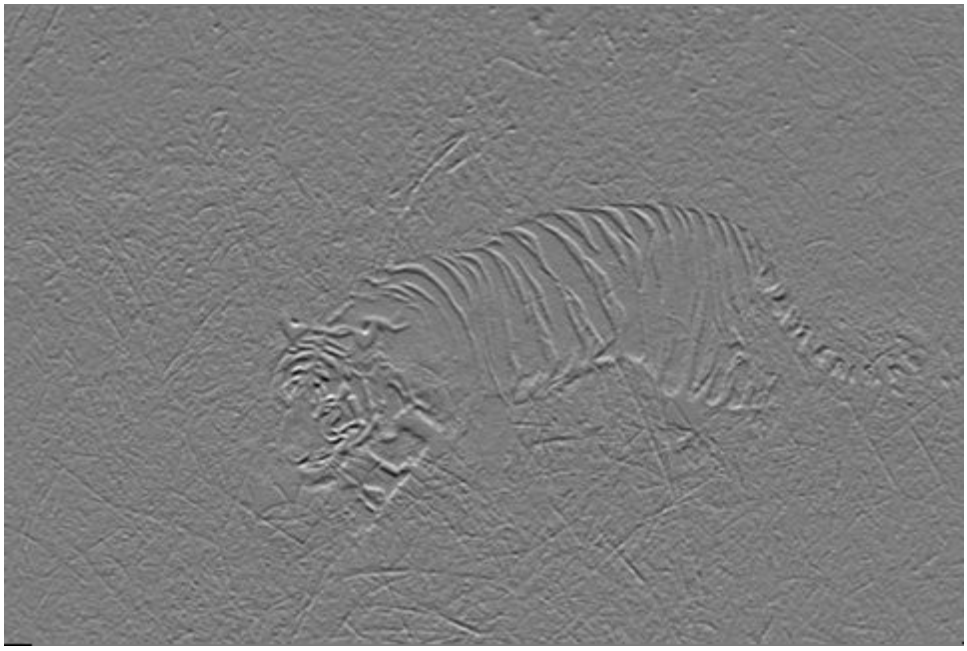


Pig Sobel Output

Tiger Sobel Output

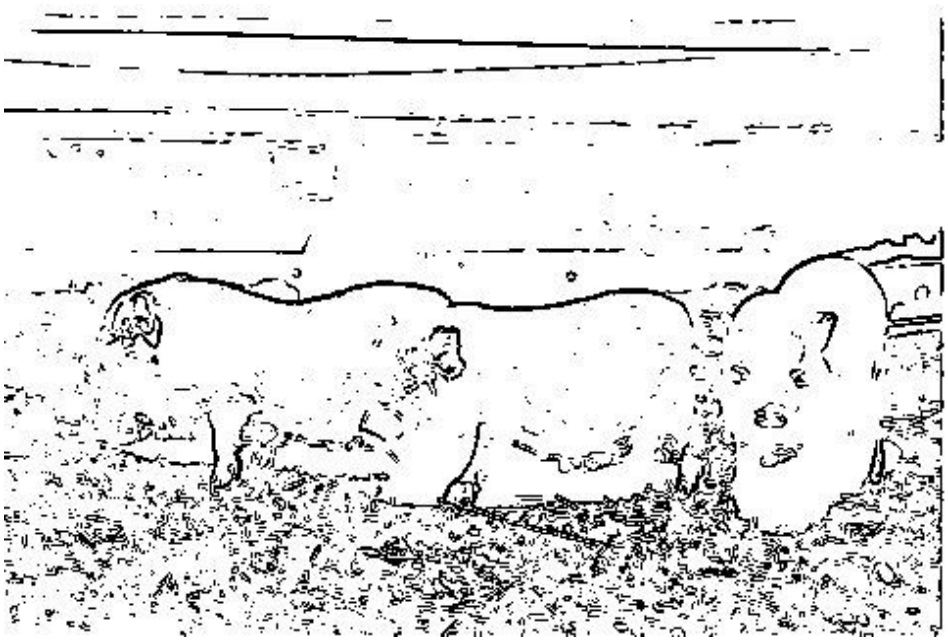


X_gradient Edge Map

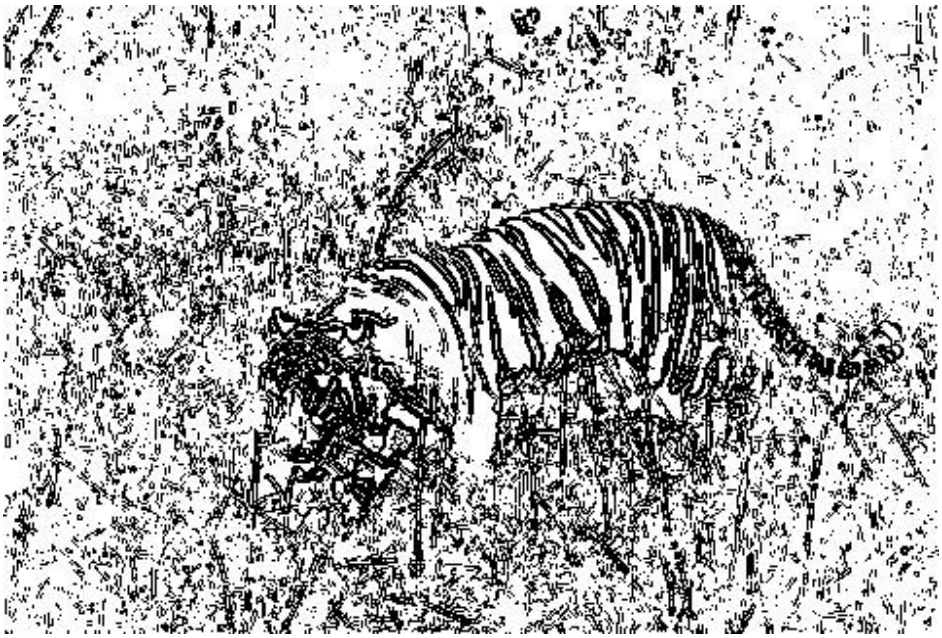


Y_gradient Edge Map

Threshold of 90%:



Threshold of 70%:



Discussion:
Threshold of 80% gives a good output. As we increase threshold to 90, important info is lost. And for 70, too many textures are highlighted.
We can see x_ gradient detects vertical edges, y_gradient detects horizontal edges.
We can see that Sobel is very sensitive to noise and texture patterns.
It cannot detect edges with less variation of intensities. (Since pig colors were same, it couldn't identify where one pig ends and other begins.)
Also, Boundaries are very thick.

b) Canny Edge Detector
Motivation:
Sobel had various drawbacks like thick edge, sensitive to noise which were taken care of while designing Canny edge detector. Canny uses second order derivative which is more stable than first order derivative.
Approach:

- Filter image with derivative of Gaussian.
- Find magnitude and orientation of gradient.

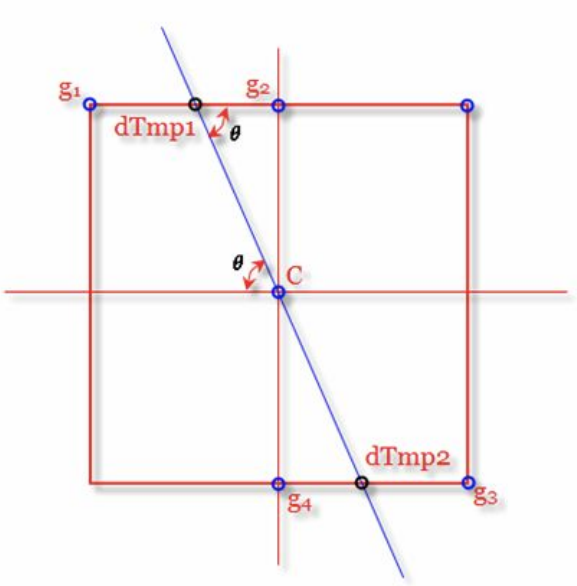
Magnitude $\quad\quad$ Orientation

$$|\nabla f| = \sqrt{g_y^2 + g_x^2} \qquad \theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

Gx and Gy can be found by matrix discussed in Sobel filter.

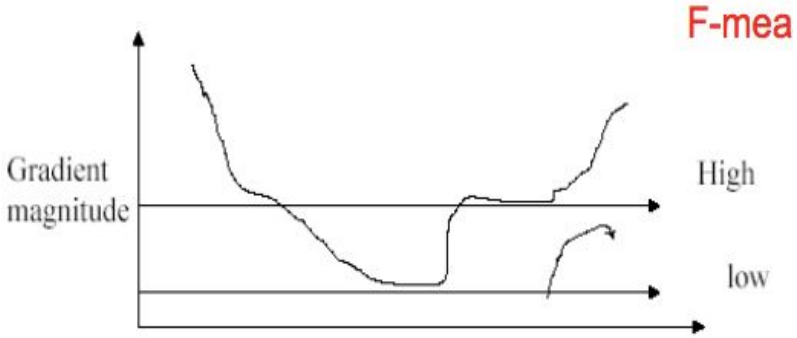
- Non-maximum Suppression (NMS)

Suppress all the gradient values to 0 except the local maxima



Preserve gradient magnitude at pixel C, if its value is larger than that of C1 and C2 (by interpolation).
Otherwise, suppress its value to be 0

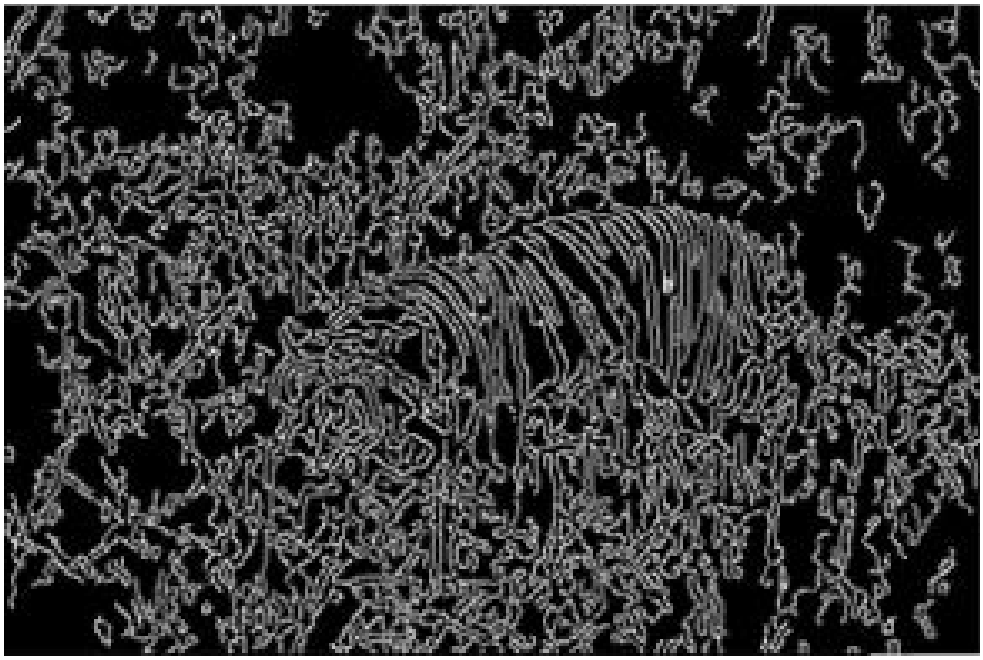
- Double Thresholding (Hysteresis Thresholding)



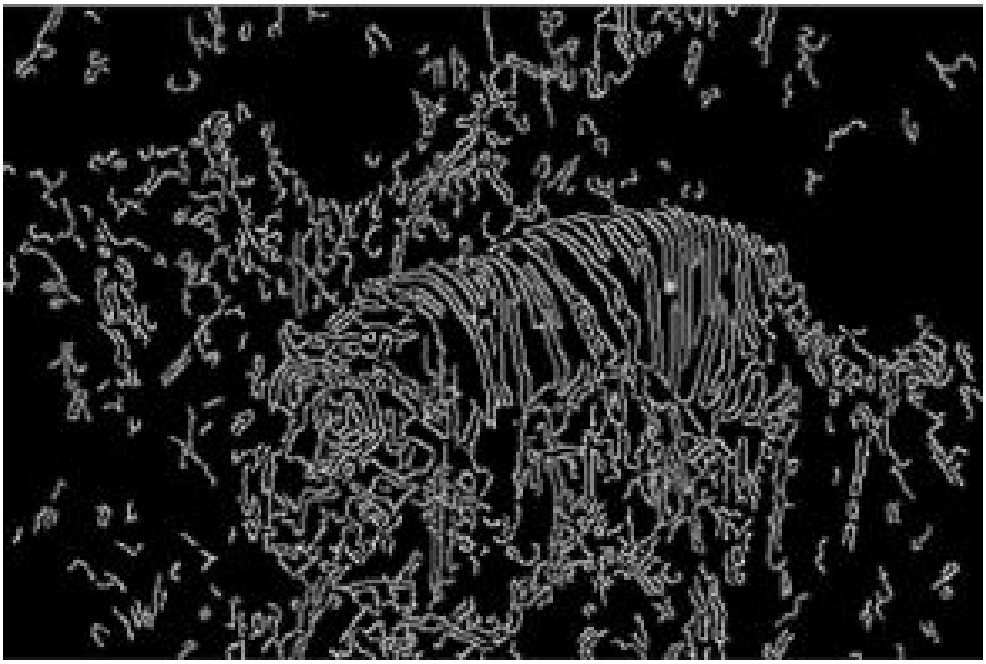
• If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge
• If a pixel gradient value is below the lower threshold, then it is rejected.
• If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.
• Canny recommended a upper:lower ratio between 2:1 and 3:1.

Experimental Results:

Tiger Outputs:



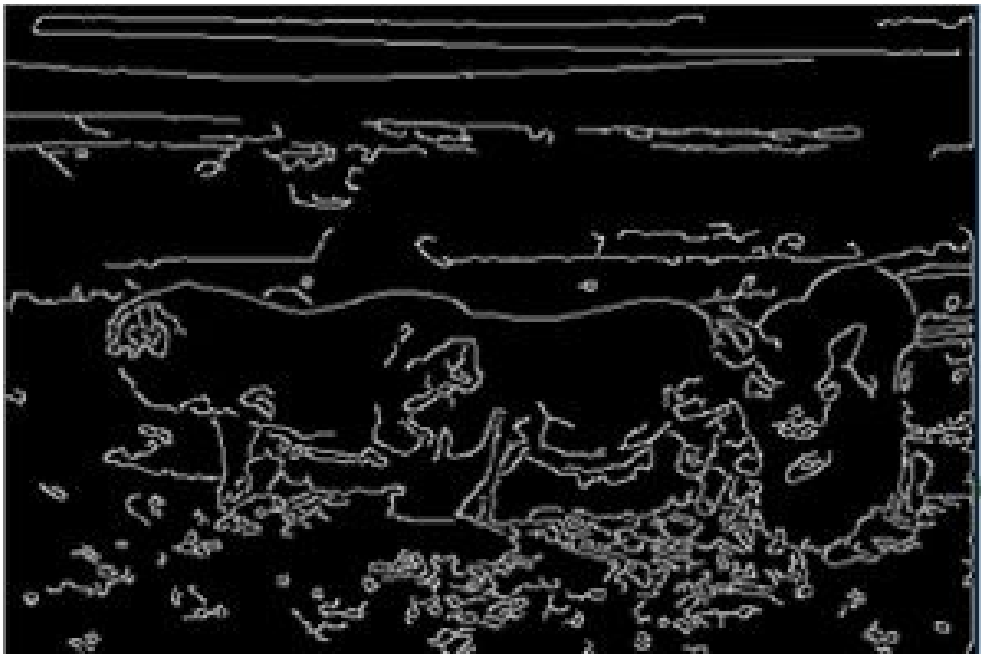
Lower threshold: 30, Upper threshold: 90, Ratio:3:1



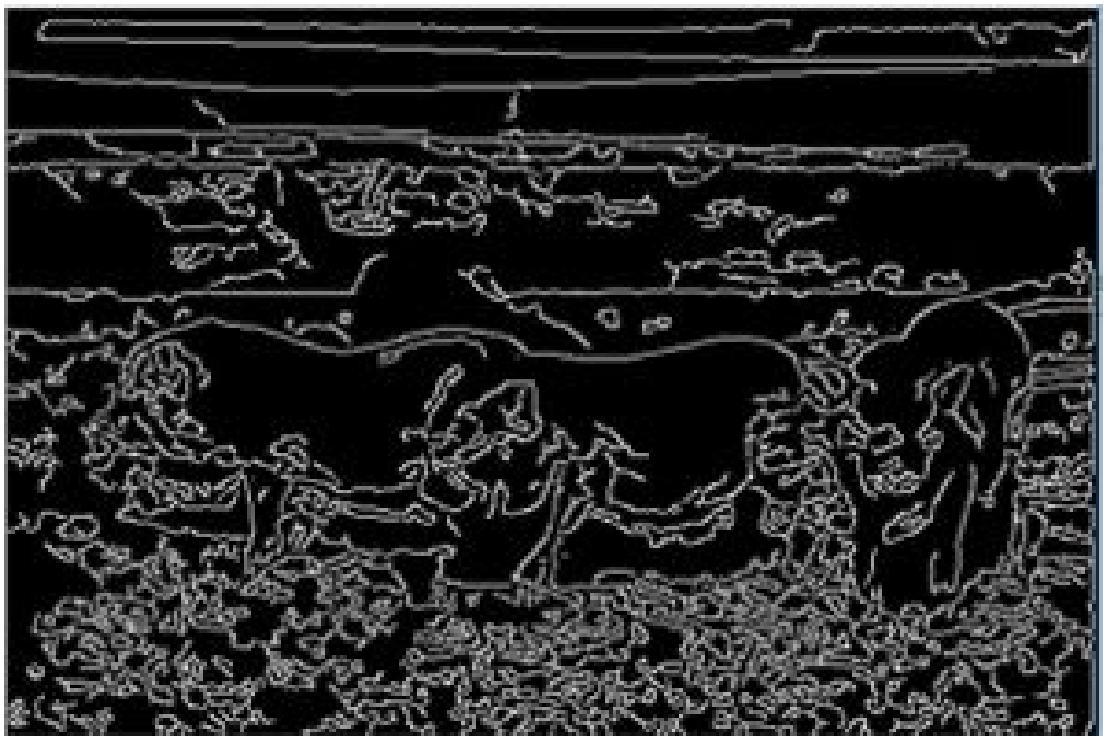
Lower threshold: 50, Upper threshold: 100, Ratio:2:1

Lower threshold: 50, Upper threshold: 125, Ratio:2.5:1

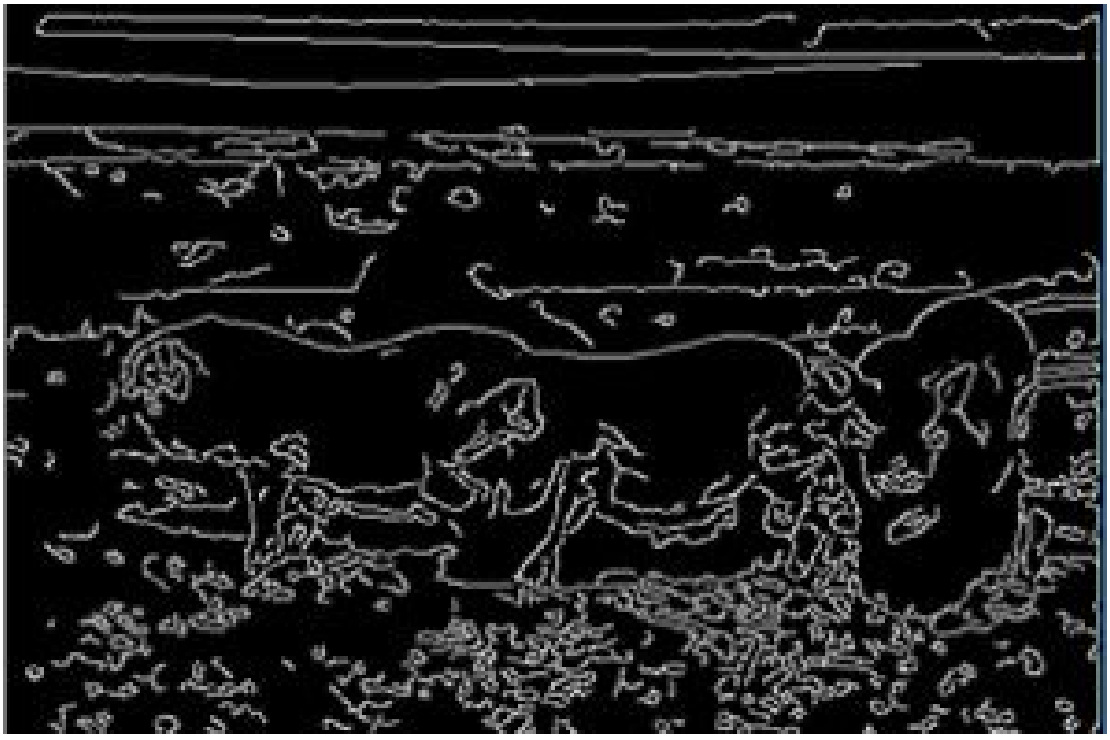
Pig Outputs:



Lower threshold: 50, Upper threshold: 125, Ratio:2.5:1



Lower threshold: 50, Upper threshold: 100, Ratio:2:1

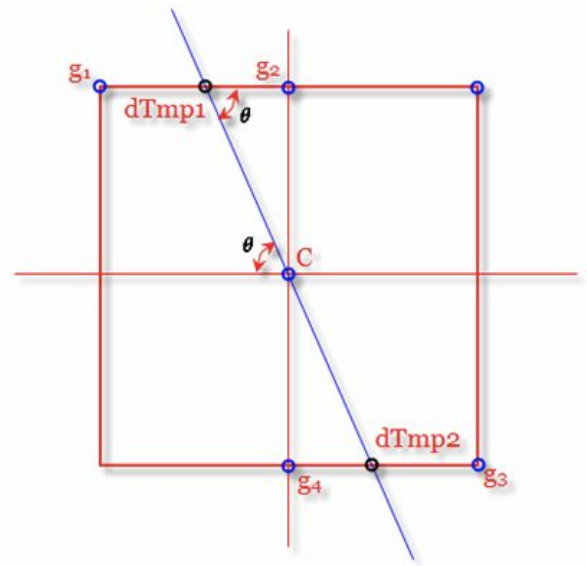


Lower threshold: 30, Upper threshold: 90, Ratio:3:1

Discussion:

1. Explain Non-maximum suppression in Canny edge detector

NMS was introduced to reduce the thickness of boundary. First we find the orientation of the boundary as discussed in approach.



Then, we compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.

If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction, the value will be preserved. Otherwise, the value will be suppressed.

2. How are high and low threshold values used in Canny edge detector?

(Described in Approach: Double Thresholding)

Too high threshold value can miss important data.

Too low threshold value falsely identifies irrelevant information.

Ratio of 3:1 highlighted unnecessary textures as edges. 2:1 misses some information without much change in shade eg, face of pig. 2.5:1 works the best for both tiger and pig.

c) Structured Edge:

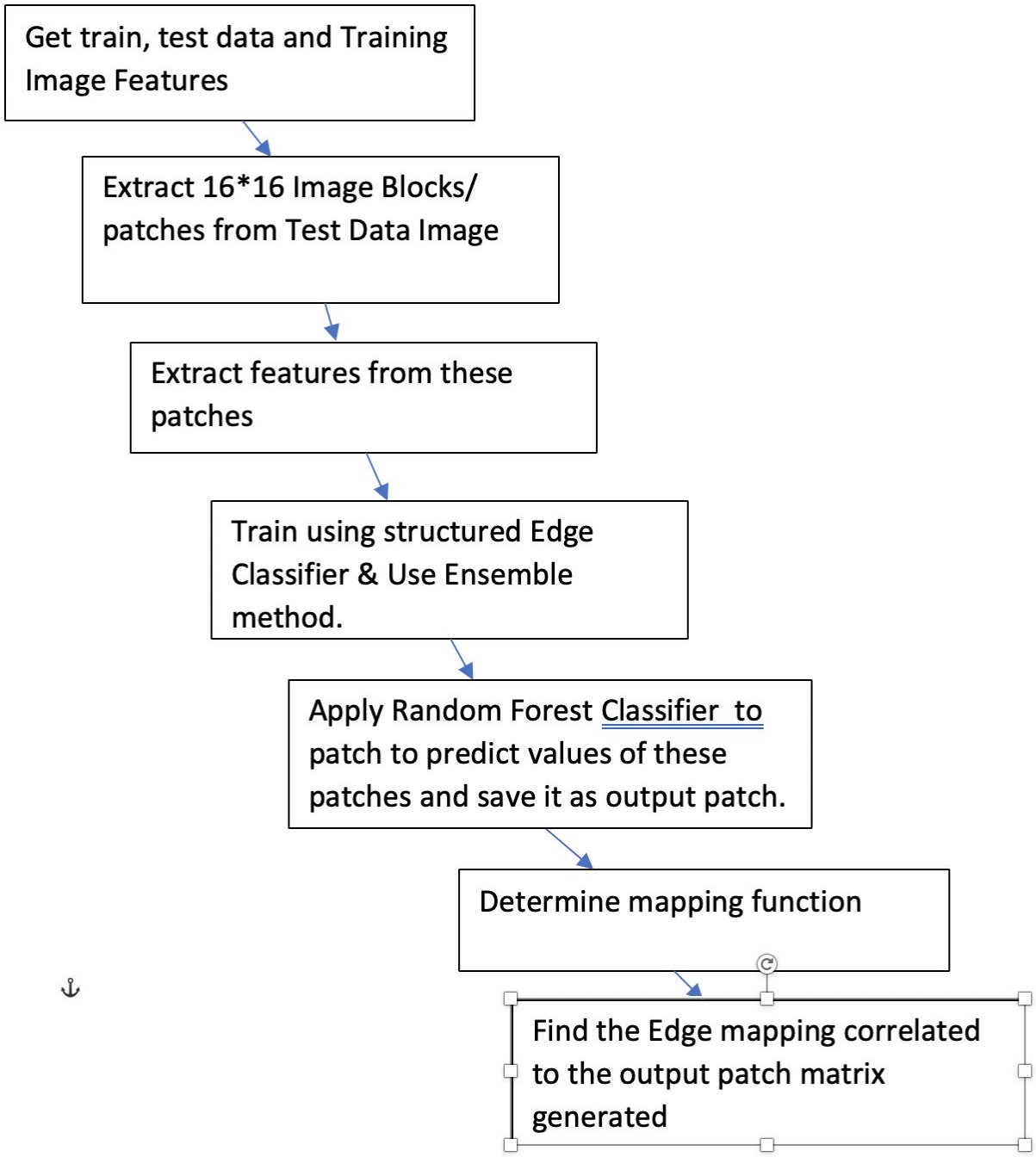
Motivation:

Structured Edge is a data driven method.

Approach:

Used edgeDemo.m

Summarize SE with a flow chart and explain it



Load the labelled training data, test data and given training edge features.

Now, to detect edge from the test pictures, divide the picture into blocks of size 16*16. Extract features from these pictures. Train using Structured Edge Classifier and use uncorrelated trees for ensemble. Apply Random Forest to predict values of these test patches. Determine mapping function and find Edge mapping correlated to the output patch matrix.

2. The Random Forest (RF) classifier is used in the SE detector. The RF classifier consists of multiple decision trees and integrate the results of these decision trees into one final probability function. Explain the process of decision tree construction and the principle of the RF classifier.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (*B* times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

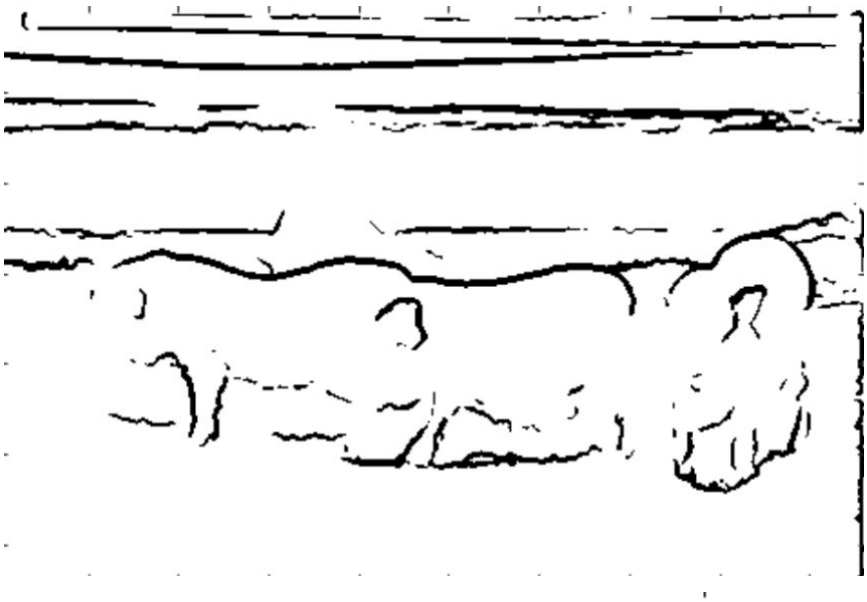
or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of features. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

Experimental Results:

Threshold: 0.85



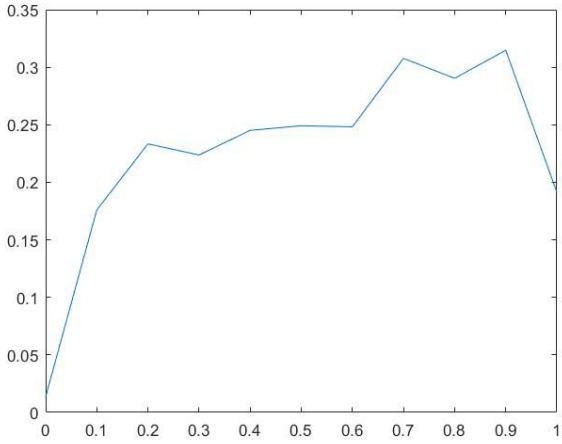
Threshold: 0.95

Discussion:
Performs better than Canny and Sobel
Uses less storage space.
As threshold value increases, edge becomes thicker. For smaller threshold, edge is thinner

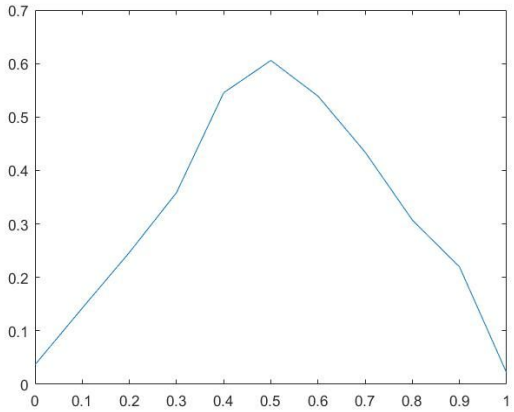
d) Performance Evaluation

Used edgsEvalImg

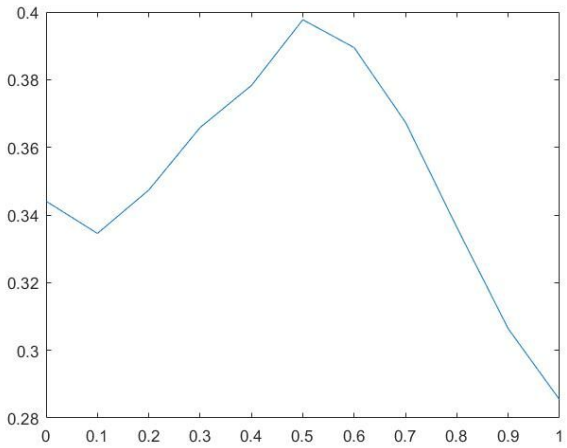
F-values:



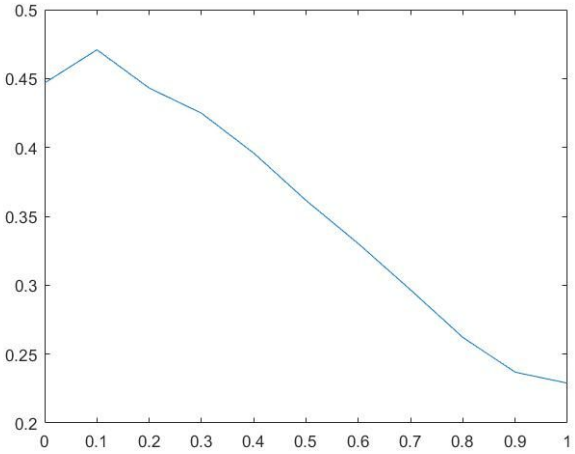
Pig sobel



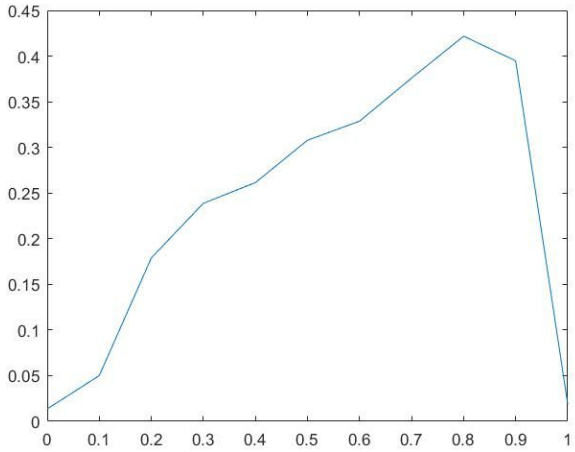
tiger sobel



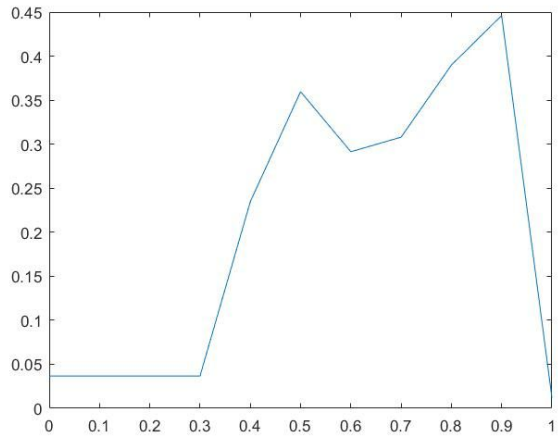
Pig Canny



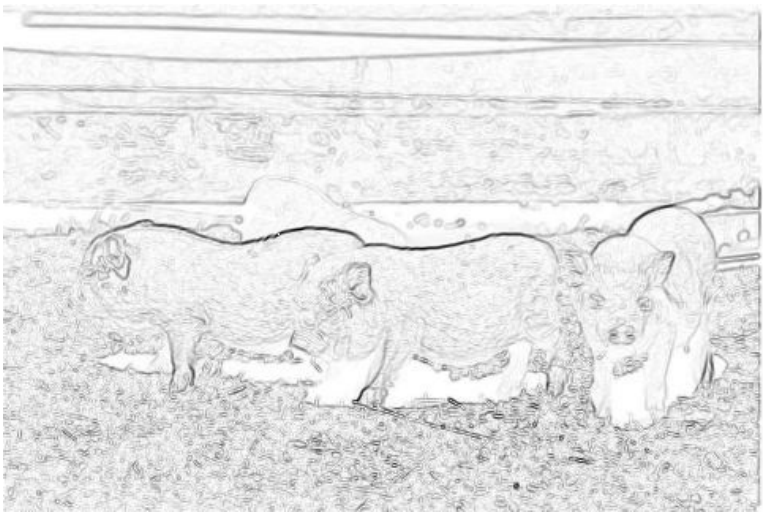
Tiger Canny



Pig Structured Edge

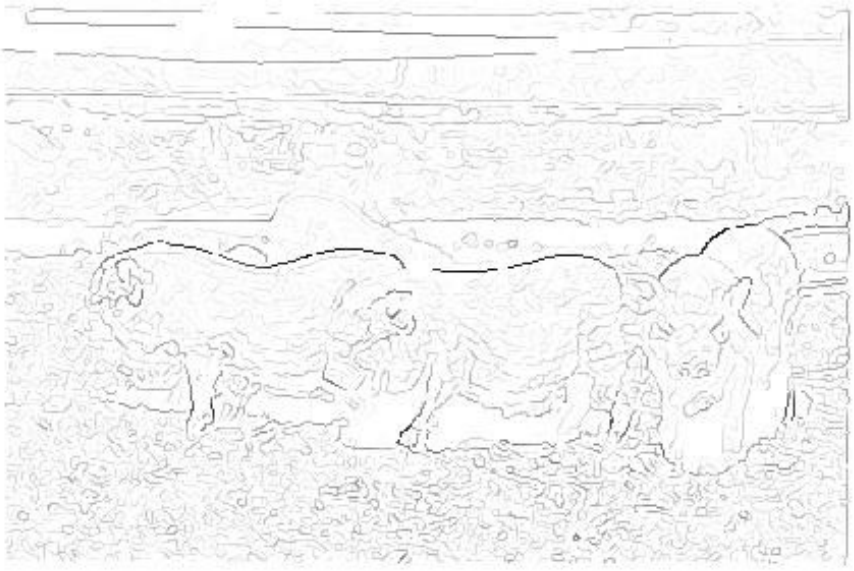


Tiger Structured Edge



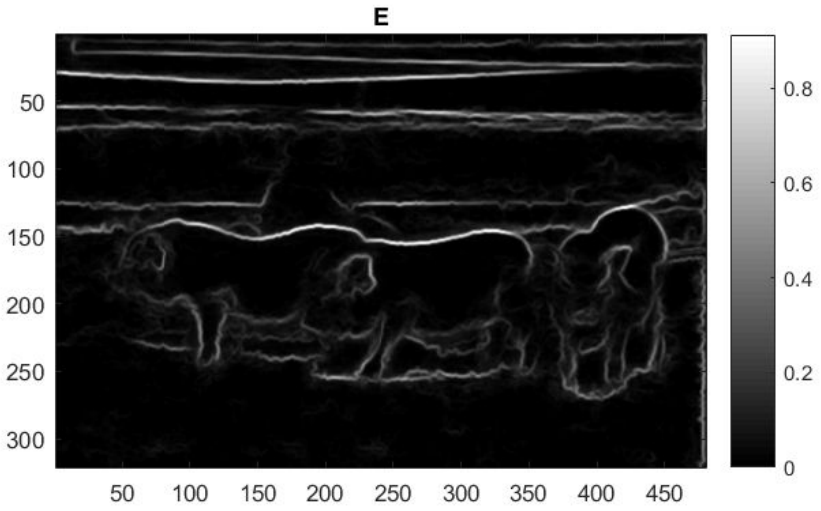
Probability map of pig sobel

Probability map of tiger sobel

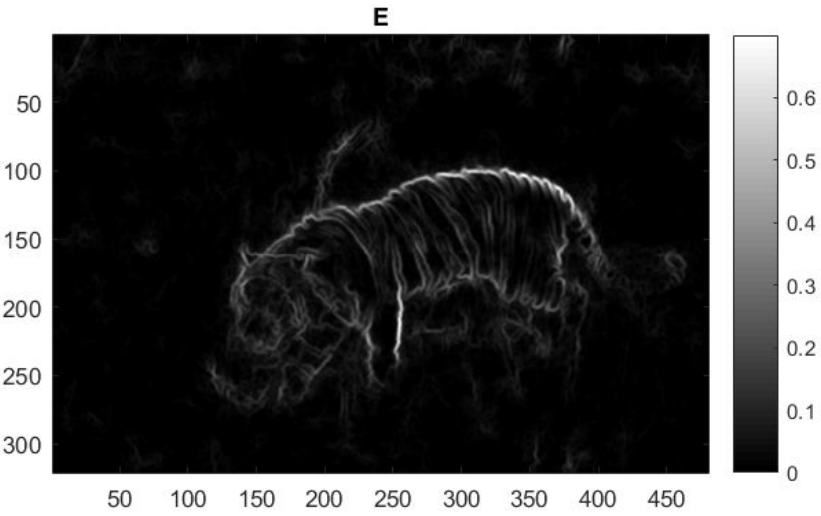


Probability map of pig canny

Probability map of tiger canny



Probability map pig Structured Edge



Probability map tiger Structured Edge

Calculate the precision and recall for each ground truth (saved in .mat format) separately using the function provided by the SE software package and, then, compute the mean precision and the mean recall. Finally, calculate the F measure for each generated edge map based on the mean precision and the mean recall. Please use a table to show the precision and recall for each ground truth, their means and the final F measure.

Structured Edge:

Pig:

Ground Truths	Mean Recall	Mean Precision	Mean F-Measure
Ground Truth 1	0.0110	0.1240	0.0167
Ground Truth 2	0.1491	0.2557	0.1865
Ground Truth 3	0.2099	0.1767	0.1897
Ground Truth 4	0.2203	0.1789	0.1985
Ground Truth 5	0.4458	0.2175	0.2578

Best F: 0.4733

Tiger

Ground Truths	Mean Recall	Mean Precision	Mean F-Measure
Ground Truth 1	0.0179	0.0146	0.0074
Ground Truth 2	0.0263	0.0783	0.0912
Ground Truth 3	0.2438	0.2745	0.2956
Ground Truth 4	0.1364	0.3989	0.1456
Ground Truth 5	0.1456	0.1230	0.1380

Best F: 0.5770

The F measure is image dependent. Which image is easier to a get high F measure - Tiger or Pig? Please provide an intuitive explanation to your answer.

As seen above, Tiger has better F measure. One reason is because of more color variation.

Discuss the rationale behind the F measure definition. Is it possible to get a high F measure if precision is significantly higher than recall, or vice versa? If the sum of precision and recall is a constant, show that the F measure reaches the maximum when precision is equal to recall.

We have a tradeoff between precision and recall. But we don't want to compromise on one, so we use F measure which is high only when both are high. Yes it is possible to get a high F measure if precision is significantly higher than recall, or vice versa. But, the difference shouldn't be much or value starts to drop.

Now when,
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 Half-toning

Many image rendering technologies only have binary output. For example, printers can either "fire a dot" or not. Halftoning is a method for creating the illusion of continuous tone output with a binary device. Effective digital halftoning can substantially improve the quality of rendered images at minimal cost. [1]

Approach:
a) Dithering

The ordered dither algorithm compares the gray level of a pixel to a level in a dither matrix. Depending on this comparison, the pixel is set to either black or white. The matrix contains threshold values between 0 and 255. The size of the matrix and the arrangement of the values have an important effect on the dither process. When a picture is larger than the dither matrix (and it usually is), the matrix is tiled over the image. Every pixel in the source image is compared to only one level in the dither matrix. Neighboring pixels have no effect on the dither result.

1.Random thresholding
Motivation:
Approach:

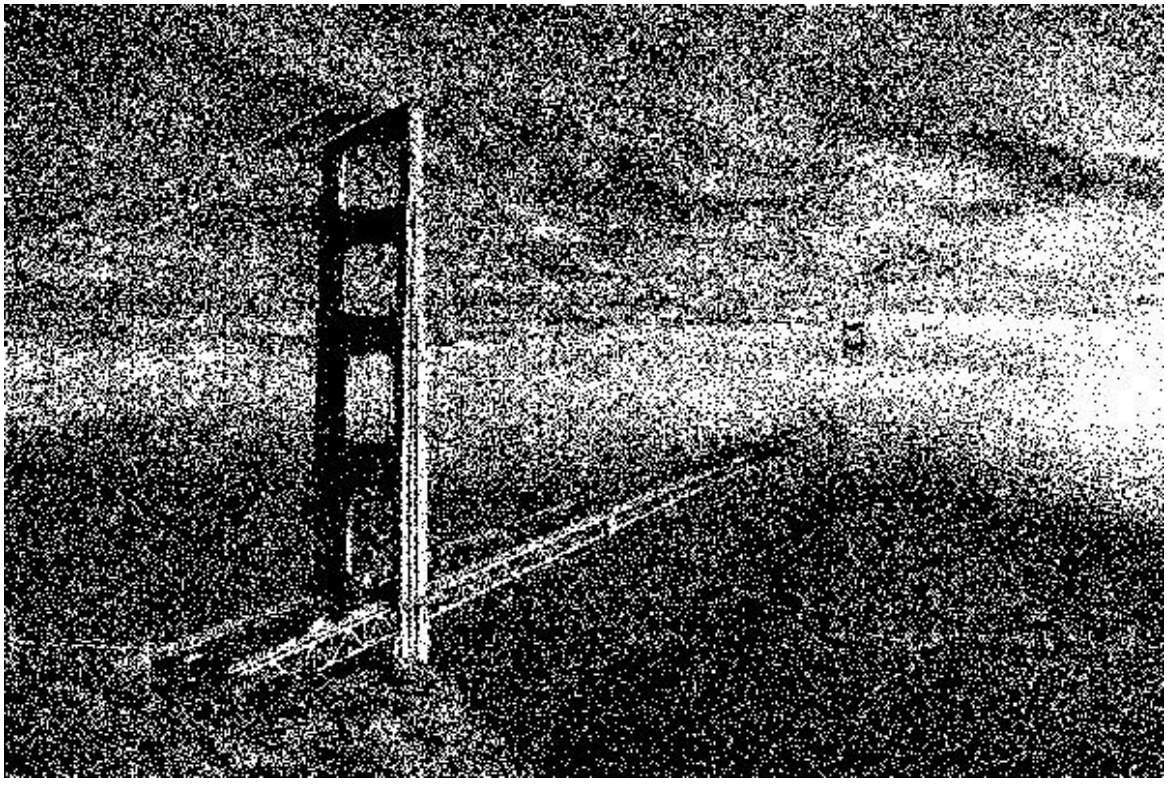
- For each pixel, generate a random number in the range 0 ~ 255, so called $rand(i, j)$
- Compare the pixel value with $rand(i, j)$. If it is greater, then map it to 255; otherwise, map it to 0,

  I.e.

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

Experimental Results:

Original Image



Output Image

Discussion:

It is very simple to describe and implement.

While random dither adds a lot of high-frequency noise to a picture, it is useful in reproducing very low-frequency images where the absence of artifacts is more important than noise. For example, a whole screen containing a gradient of all levels from black to white would actually look best with a random dither. In this case, ordered dithering would produce diagonal patterns, and error dispersion would produce clustering.

For efficiency, you can take the random number generator "out of the loop" by generating a list of random numbers beforehand for use in the dither. Make sure that the list is larger than the number of pixels in the image or you may get artifacts from the reuse of numbers. The worst case would be if the size of your list of random numbers is a multiple or near-multiple of the horizontal size of the image, in which case unwanted vertical or diagonal lines will appear.

2. Dithering Matrix

Motivation:

Random threshold creates grainy output with high- frequency noise. To reduce this, Dithering Matrix tries to make a threshold matrix which reduces this noise.

Approach:

Dithering parameters are specified by an index matrix. The values in an index matrix indicate how likely

a dot will be turned on. For example, an 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. This index matrix is a special case of a family of dithering matrices first introduced by Bayer. The Bayer index matrices are defined recursively using the formula:

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

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:

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

where N^2 denotes the number of pixels in the matrix. Since the image is usually much larger than the threshold matrix, the matrix is repeated periodically across the full image. This is done by using the following formula:

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

Experimental Results:

Original

n=2

n=8

n=32

Discussion:

There are two major issues with ordered dithering. First, important **visual artifacts** are seen. Even Bayer ordered dithering causes weird cross-hatch pattern artifacts on some images. Second, dithering matrices do not depend on the original image and thus **do not take input data into account**: high frequency features in the image are often missed and, in some cases, cause even worse artifacts.
As n increases, more details are captured, but grid patterns increase. In smaller n value, smoothness is lost and we might see false contouring.

b) Error Diffusion:

Motivation:

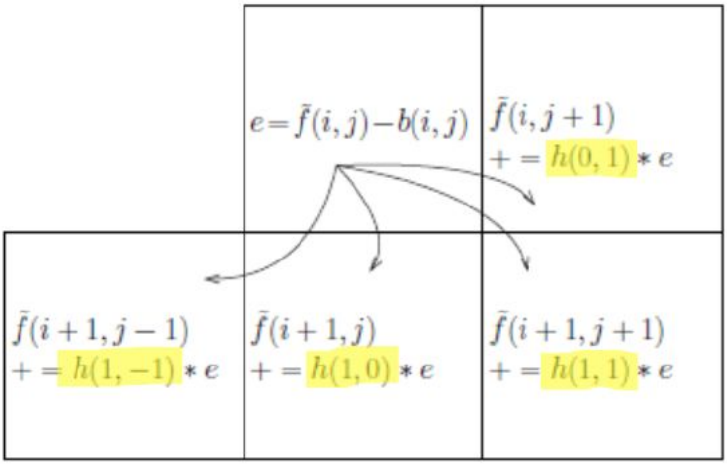
The idea behind error diffusion is to compute the error caused by thresholding a given pixel and propagate it to neighbour pixels, in order to compensate for the average intensity loss or gain. It is based upon the assumption that a slightly out-of-place pixel causes little visual harm.

Approach:

1. Initialize $\tilde{f}(i,j) \leftarrow f(i,j)$
2. For each pixel:
    - Binarize:

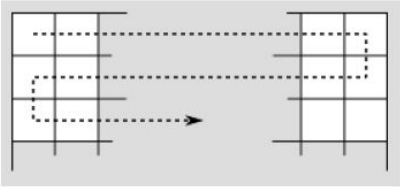
$$b(i,j) = \begin{cases} 255 & \text{if } \tilde{f}(i,j) > T \\ 0 & \text{otherwise} \end{cases}$$

    - Diffuse Error:



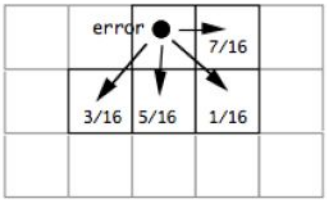
    - Scanning order :

- **Serpentine parsing**



1. Floyd-Steinberg's error diffusion with the serpentine scanning, where the error diffusion matrix is:

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



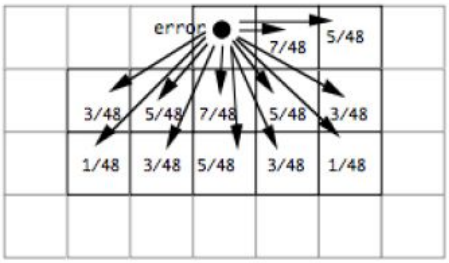
Experimental Results:

Discussion:

- Only diffuses error to 4 neighbouring pixels.
- We see very **fine grained** Dithering.
- We can also observe random white spots.

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

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



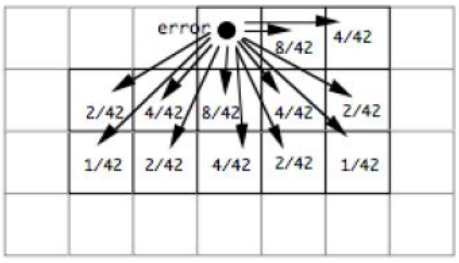
Experimental Results:

Discussion:

- One step further than Floyd Steinberg, JJN diffuses error to 12 neighbors.
- Image is still grainy but appears a lot better without white spots.
- It's slower than FS.

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

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



Experimental Results:

Discussion:

- Stucki too diffuses error to 12 neighbors but is slightly faster than JJN.
- Image appears grainy.
- But is clean and sharp.

Discussion for Diffusion:

Which method do you prefer? Why?

Stucki. It gives the best output.

Error diffusion has the tendency to enhance edges in an image. This can make text in images more readable than in other halftoning techniques.

Comparison between Dithering and Error Diffusion:

Dithering

• More uniform dot placement and a limited spatial bandwidth
• Uniform dot placement without worms defects or coarse noise pattern
• Grainer than error diffusion
• Blurred lines because of the limited bandwidth
• Lack of thin details , not well defined contour, and not unpleasant patterns
• Faster than Error Diffusion

Error Diffusion:

• Broad spatial bandwidth and a pleasant dot placement, but having the worms defect
• Very pleasant for dot distribution, not appearing worms and coarse noise pattern

• Unpleasant worms defect
• Coarse noise patterns
• Unpleasant patterns in the gray scale
• Well defined thin lines
• Good thin details and coarse noise pattern

c) Color Halftoning with Error Diffusion

1. Separable Error Diffusion:
Motivation:
The idea of error diffusion for monochrome is just extended to do Color Halftoning for 3 seperate channels.
Approach:
Above Error Diffusion using Floyd Steinberg is repeated exactly for 3 channels separately.
Experimental Results:

Original Image

Output halftoned image

Discussion:

- We see sharp edges.
- We also see random dots from one channel dominating in the image and is directly visible due to sudden change in color.
- Computational Complexity increases three times than monochrome image.

2.MBVQ-based Error diffusion

Motivation:

Relevant to the problem at hand is the fact that the human visual system is more sensitive to changes in brightness than to changes in the chrominance, which average at much lower frequencies. Thus we arrive at the Minimal Brightness Variation Criterion for halftoning of solid color patches.

Approach:

For each pixel (i; j) in the image:

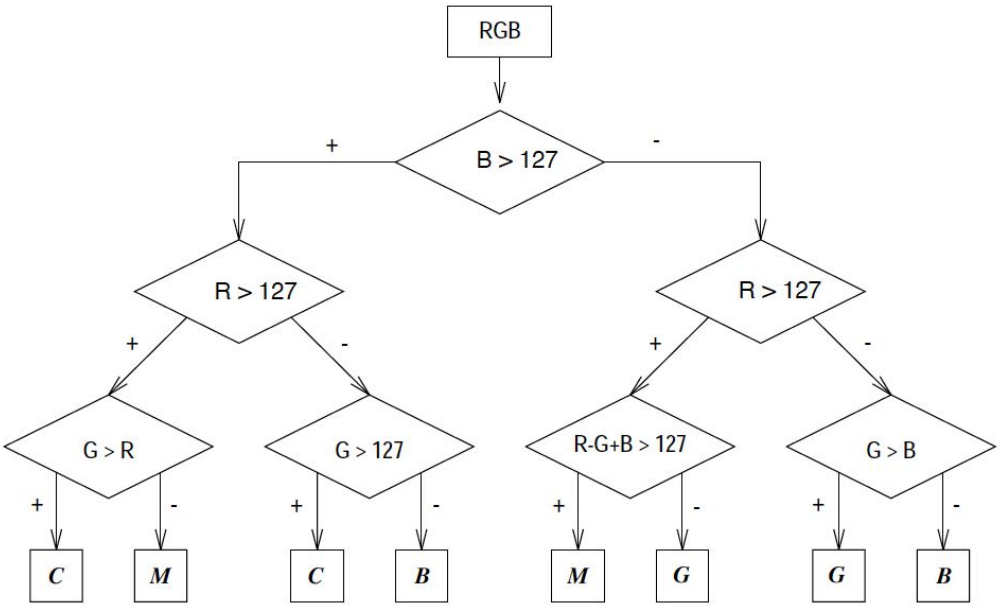
1. Determine MBVQ(RGB(i; j)).

```
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 ∈ MBVQ which is closest to RGB(i; j)+e(i; j).

Eg. for CMGB,



3. Compute the quantization error RGB(i; j) + e(i; j) - v.
4. Distribute the error to “future" pixels using Floyd Steinberg.

Experimental Results:

Original Image

Output haltoned image

Discussion:

1) Describe the key ideas on which the MBVQ-based Error diffusion method is established and give reasons why this method can overcome the shortcoming of the separable error diffusion method.

- Monochrome halftone algorithms are carefully designed to reduce visible artifacts. One of the most important factors producing those artifacts is the variation in the brightness of the dots. In binary halftones (Black and White), this factor cannot be mitigated.
  To produce a good color halftone one has to place colored dots so that the following specifications are optimally met:
  (1) The placement pattern is visually unnoticeable.
  (2) The local average color is the desired color.
  (3) The colors used reduce the notice-ability of the pattern.
  The first two design criteria are easily carried over from monochrome algorithms.
  However, the third cannot be satisfied by a simple Cartesian product generalization of monochrome halftoning which is used in separable error diffusion.
- It is known that given a color in the RGB cube, it may be rendered using the 8 basic colors located at the vertices of the cube. Actually, any color may be rendered using no more than 4 colors, different colors requiring different quadruples. Moreover the quadruple corresponding to a specific color is,in general, not unique (in a linear color space, any quadruple whose convex hull contains the desired color will do).
- The major difference between separable Error Diffusion and Color Diffusion is in step (2), where the algorithm looks for the closest vertex in the MBVQ of the color, as opposed to the closest of the eight vertices of the cube. This step takes
- This, takes care of the third shortcoming mentioned in separable Error Diffusion.

2) Compare the MBVQ output with that obtained by the separable error diffusion method.

- Image is much crispier with MBVQ error diffusion.
- Random color dots don't appear in middle of a smooth texture.
- Image looks brighter and much more visually appealing.
- Edges look sharper due to good contrast.

Reference:
[1]https://engineering.purdue.edu/~bouman/ece637/notes/pdf/Halftoning.pdf
https://www.visgraf.impa.br/Courses/ip00/proj/Dithering1/random_dithering.html
[2] https://pdfs.semanticscholar.org/6dac/eb63ec587522f9e5e230ee5ede4161eefc7e.pdf
[3]http://shodhganga.inflibnet.ac.in/bitstream/10603/176027/9/09_chapter%201.pdf
[4]https://pdfs.semanticscholar.org/c67f/37a2ab36bab46bb632b65dc8dc3866f7c80e.pdf