EE 569 HOMEWORK 2

Royston Marian Mascarenhas

PROBLEM A

Sobel edge detector

(a)

1. **Abstract and Motivation:**

   In digital image, edges can be described as curved lines composed of the points where brightness change sharply. They are the boundaries of the objects and can be used to capture important information in image. The Sobel edge detector is a first order edge detection filter which takes the derivate of Gaussian and uses the gradient magnitude to determine the edges.

2. **Approach and Procedures:**

   The three steps involved are:

   1. Convolve X Gradient and Y Gradient filters which approximate first order Gaussian derivate with pixels through window intervals to find X and Y Gradient.
   2. Find the magnitude of the gradient using the following formula:

      $$mag = sqrt(x^2 + y^2)$$

      Normalize the magnitude to pixel intensity range.
   3. Sample the top n% edges by thresholding the output image at (1-n)%

   The Sobel filter mask is given by

-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

   Gx detects vertical edges and Gy detects horizontal edges.

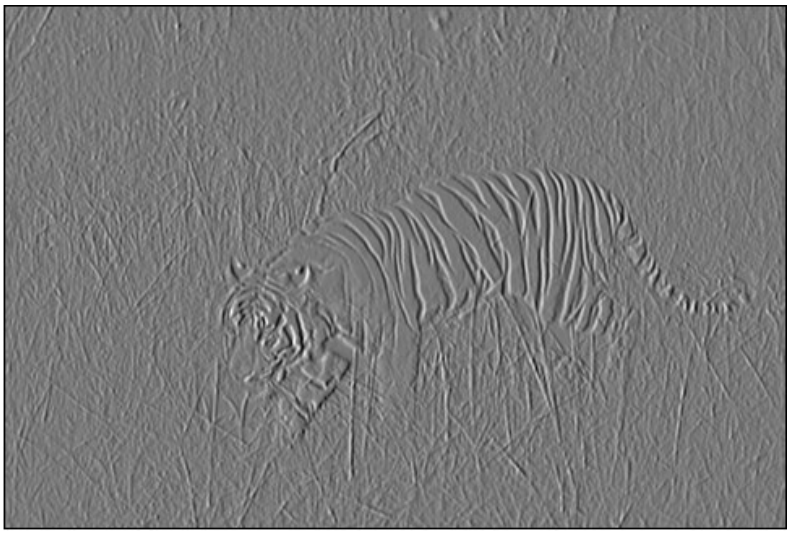
   Additional Setup:

   - Boundary Extension
   - RGB to Grayscale conversion

3. **Results:**

Tiger - grayscale

Tiger – Normalized X gradient (below)



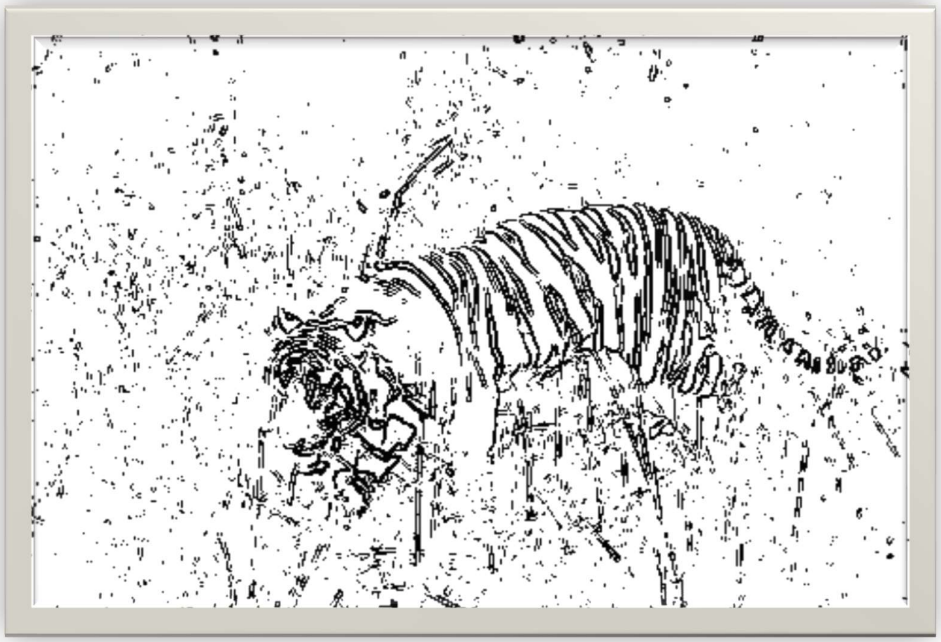
Tiger – Normalized Y gradient

Tiger – Normalized Magnitude

Threshold is 70% (Top 30%) edges considered)

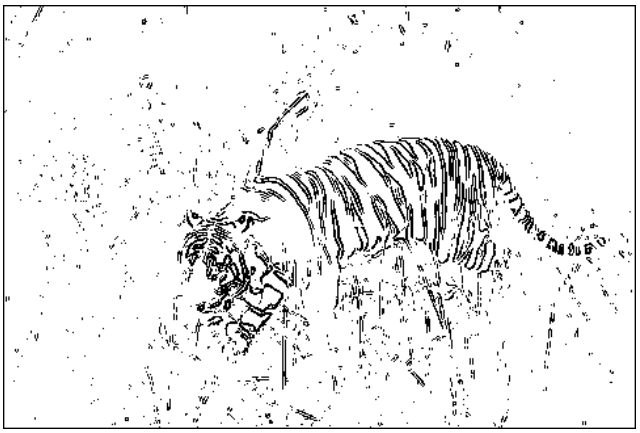
Threshold is 77%

Threshold is 88%

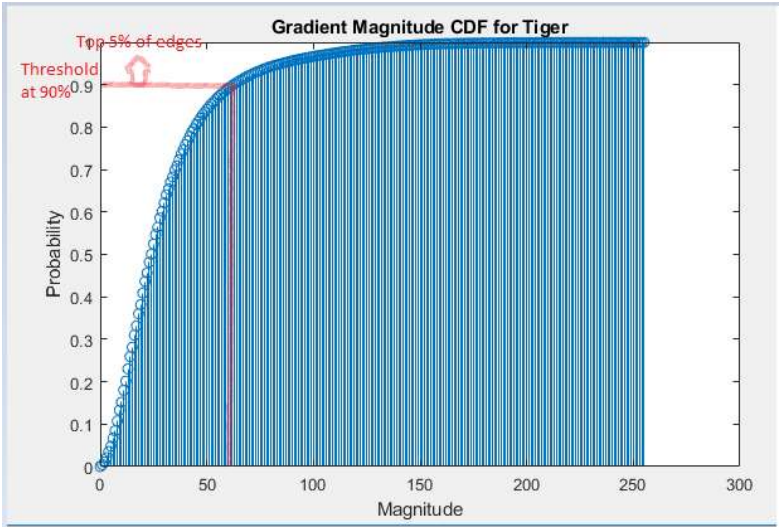


Threshold is 90%

Threshold at 93%



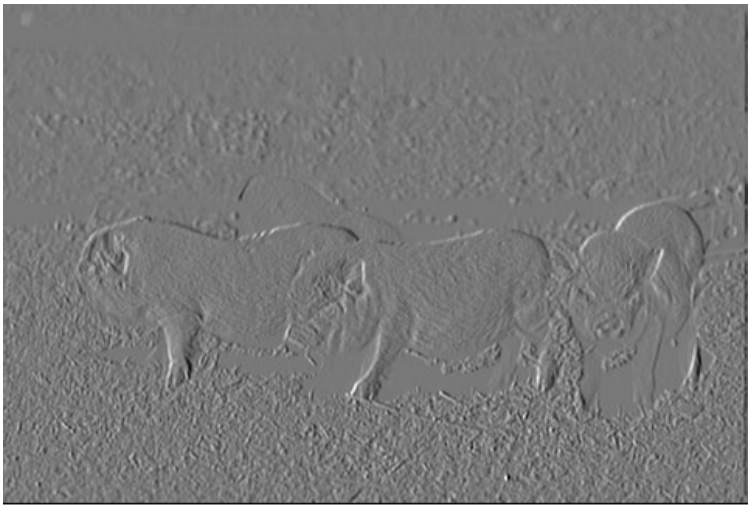
Threshold at 95%



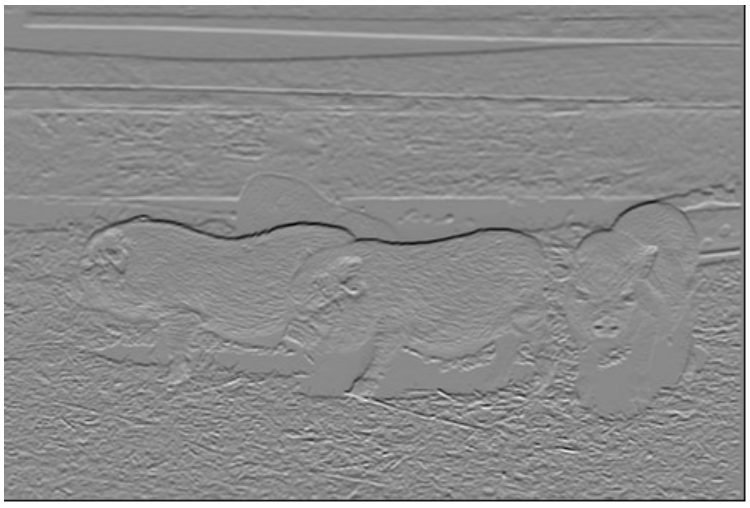
Gradient Magnitude CDF for Tiger with a sample threshold

Pig – Gray Image

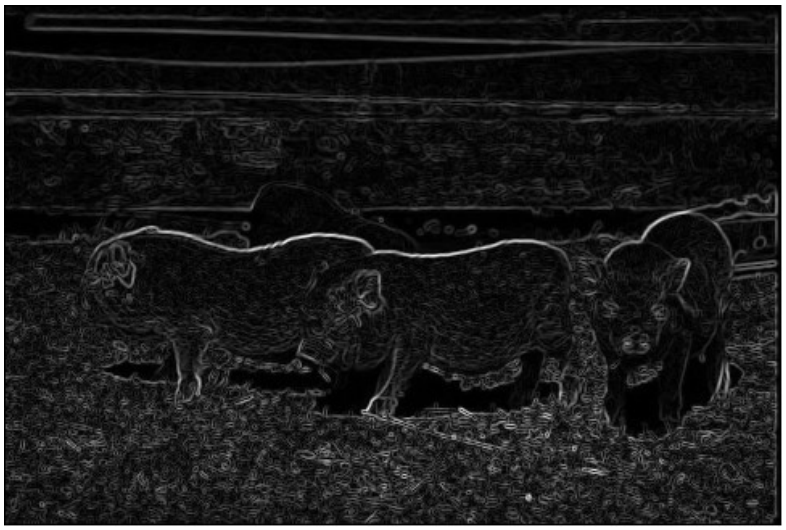
Pig – Normalized X Gradient



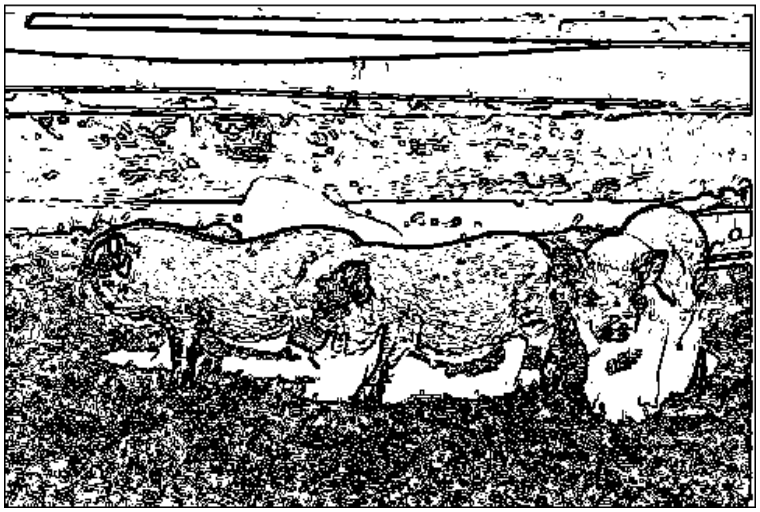
Pig – Normalized Y Gradient



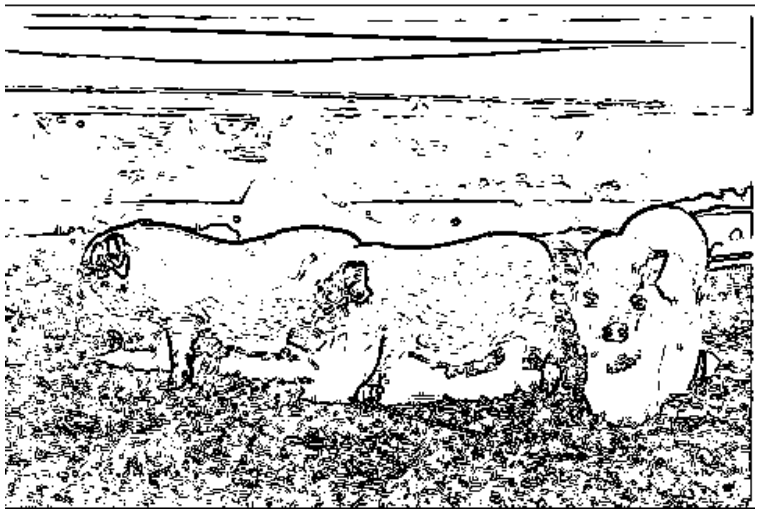
Pig - Normalized Magnitude



Threshold at 60%

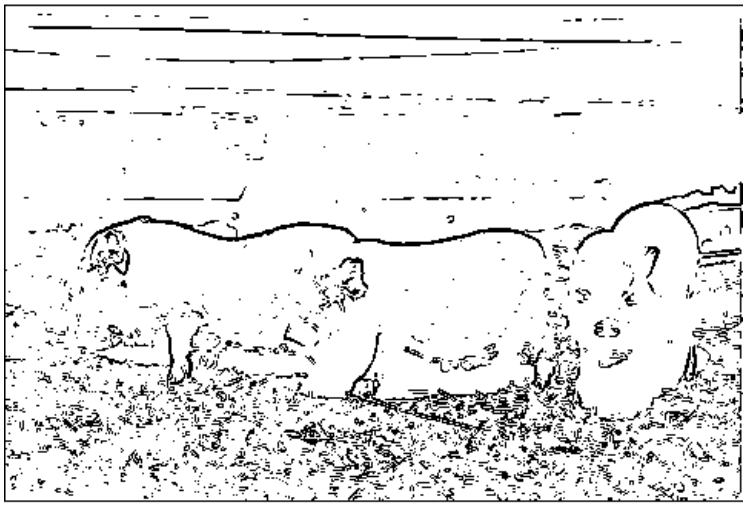


Threshold at 80%



Threshold at 85%

Threshold at 90%



4. **Discussion**:

- Edges are identified but this includes irrelevant edges as well.
- This is because the fiilter is susceptible to erroneous intensity values caused by noise.
- Edges are disconnected, as the threshold bluntly classfies strong edges and weak edges without room for inclusion for the intermediant relevant range.
- Edges are thick, improperly localized
- Best case output for tiger was 90% and for pig as well.
- Observable tiger and pig edges were on average between 55% and 93%.
- Nonetheless, relevant edges were ignored like the pig's and tiger's snout or the pig standing before the pig in the background. Unnecessary edges like grass and lawn were included in the high threshold range as well.

Canny edge detector

(b)

1. **Abstract and Motivation:**

   The Sobel edge detector does not provide the best output. The edges are thick, almost every edge is considered, even with high threshold and the edges are broken and not connected. A Sobel edge detector is also sensitive to noise. Canny edge detection aims to override these deficiencies with the help of post processing techniques.

2. **Approach and Procedures:**

   Canny edge detector consists of four major stages:

   1. Filter the image I with derivatives of Gaussian IGx and IGy.
   2. Find the magnitude and the orientations and perform normalization where required.
   3. Perform non maximum suppression on the above output to get finer edges
   4. Perform hysteresis thresholding to get connected edges.

   Answer to Problem 1 (b) 1: Non – Maximum Suppression:

   NMS is used mainly to convert a thick edge to a finer edge, thus localizing the edge to its righteous boundaries. Non-maximum suppression finds the the largest edge. After applying gradient calculation, the edge extracted from the gradient value has more than one response. Thus non-maximum suppression can help to suppress all the gradient values (by setting them to 0) except the local maxima, which indicate locations with the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:

   - 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 (i.e., 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. Otherwise, the value will be suppressed.

   In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbours in the gradient direction.

3. **Results:**

Tiger after Canny edge detection with upper threshold at 145 and lower threshold at 105. Best Case output.

Best Case output for Pig. Threshold ratio was 0.075/0.175 -> low to high 1:2

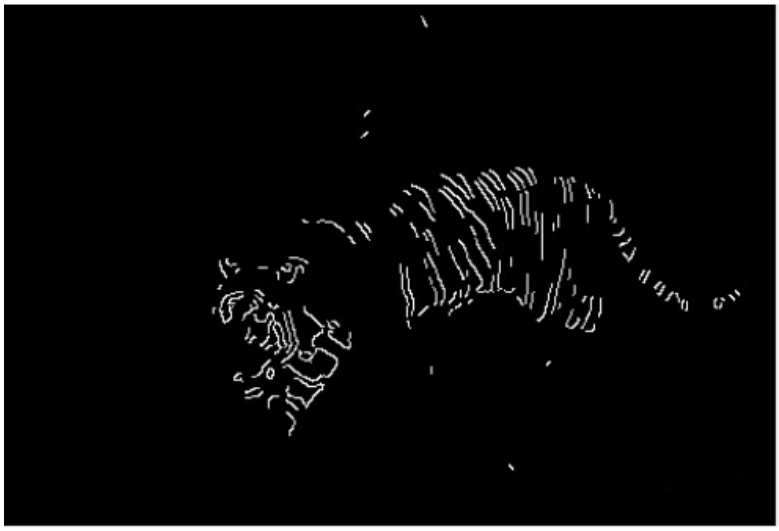
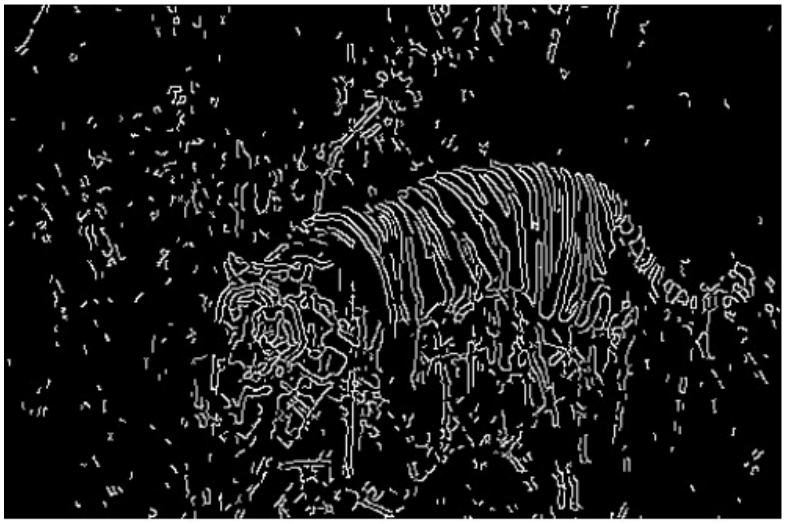


Figure (a): Effect of setting upper threshold high (explained in discussion)



Figure(b): Effect of setting lower threshold low(explained in discussion)

4. Discussion:

Answer to Problem 1 b 2:
Hysteresis Thresholding:

After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image. However, some edge pixels remain that are caused by noise and colour variation. In order to account for these spurious responses, it is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values. If an edge pixel's gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel's gradient value is smaller than the high threshold value and larger than the low threshold value, it is

marked as a weak edge pixel. If an edge pixel's value is smaller than the low threshold value, it will be suppressed.

Observations for the above answer and discussion:

As we can see above, in figure (a) the upper threshold is set high. This results in

- qualified edges which are not as strong as the strongest edges being left out
- the edges in the intermediate region getting suppressed due to absence of the above edges.
- This results in an incomplete image.

In figure (b), the threshold is set low. In this case,

- edges which are not qualified as edges but are anyway connected to stronger edges get included as edges.
- This includes high frequency noise in the image and textures which are not part of the original outline.
- This results in irrelevant details being included.

This is an improvement over the previous detector.

- Edges are connected due to hysteresis thresholding
- Thinner edges due to non maximum suppression
- Sensitivity to textures is still an issue

Structured Edge Detection

©

1. Abstract and Motivation:

The previous edge detectors do not take advantage of the inherent structure in edges. Here, edge detection is done by predicting local segmentation masks given input images. This method is computationally efficient and can determine edge maps in real time as well as produce a higher quality edge map with a high F value.

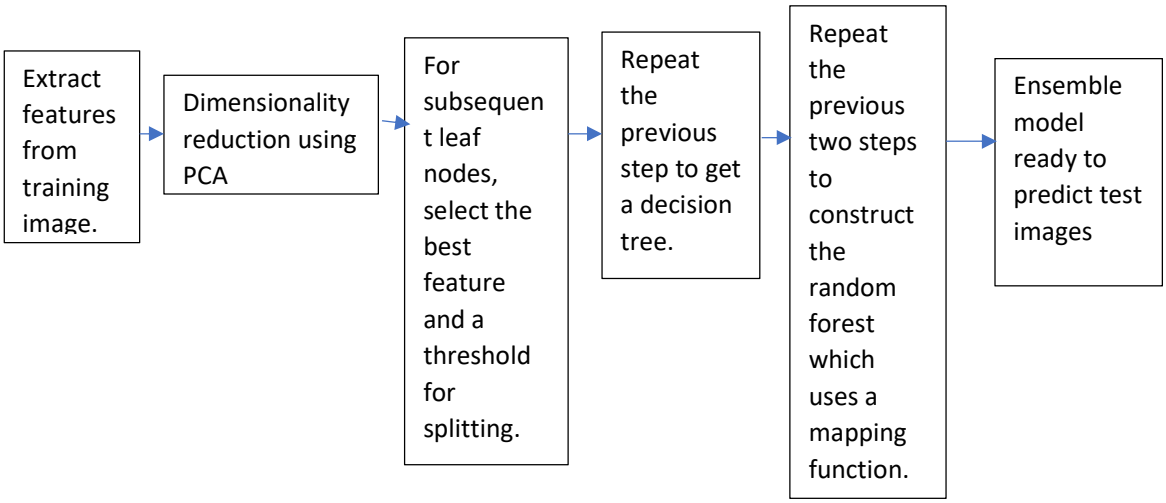
2. Approach and Procedures:

Answer to problem 1 © 1:

The SE detection method learns from data containing edge structures. A random forest frameworlk is used to capture the edge information. The approach learns decision trees which use structured labels to determine the splitting functions at each branch in the tree. The structured labels are robustly mapped to a discrete space on which standard information gain measures may be found. Each forest predicts a patch of edge pixel labels that are aggregated across the image to compute the final edge map.

We input an RGB image to the SE detector We input an RGB image into the SE detector where it computes the probability edge map (E) for the corresponding image. The algorithm is a local patch based method which extracts a s x s image patch. The probability edge map basically indicates whether a give pixel is an edge or not Given a s x s image patch (eg: s = 32 or 16), the algorithm predicts a s/2 x s/2 segmentation mask indicating segment membership for each pixel on the output probability edge map.

The input feature x is computed such that $x\epsilon\ \mathbb{R}$(s x s x k) where k are the total number of channels in each feature. We obtain two main categories of features namely Pixel lookup and Pixel differences. Other channels of feature vectors are created by adding additional information. Total number of channels by pixel lookup method total to 13. Every channel is then blurred with radius = 2 traingle filter and down sampled by a factor of 2, which would correspond to 3328. Again the triangle blur method is applied with radius = 8 to each channel and down sampled again. Here we sample all the candidate pairs and compute their differences thus generating pixel difference method. Earlier we had computed candidates using pixel lookup method. The total number of candidate features total up to be 7228. Next we use a mapping function to train the decision trees. This mapping function is denoted by π: $Y \rightarrow Z$ where Z is an intermediate space. The ensemble model is then used for combining multiple predictions obtained from numerous decision trees. The probability edge map obtained is then thresholded to obtain a binary edge map.



Answer to Problem 1 c 2:

The Principle of the random forest classifier is based on a decision tree. A decision tree *ft(x)* classifies a sample *x* belonging to *X* by recursively branching leaf or right down the tree until a left node is reached. Each node *j*, has a binary split function associated with it given by:

$$\hbar\ x, \theta 6\ \in \{0,1\}$$

Depending on the value of the split function, the node sends the input either left or right to a terminal node or a passage node. The output of a tree on an input is the prediction stored at the leaf reached by that input which might be a target label.

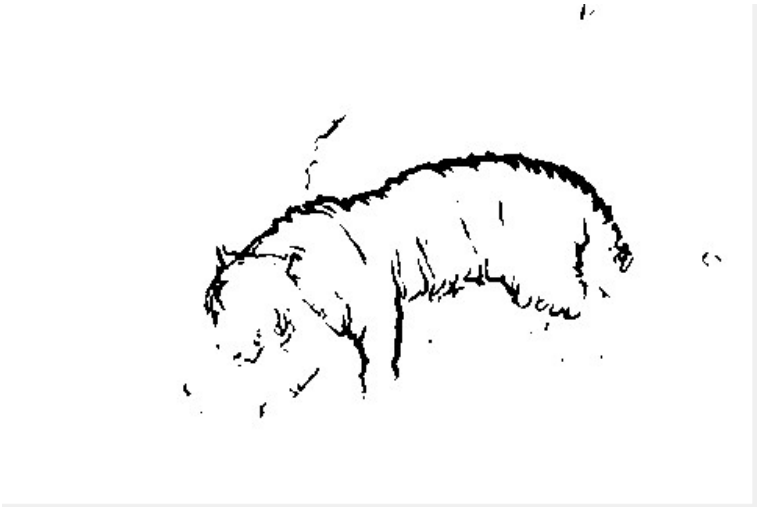
Working:

- A random forest is an ensemble of T independent trees.
- Given a sample x, the predictions from the set of trees are combined using an ensemble model into a single output.
- Choice of ensemble model is problem specific and depends on the output space,
- Common choices include majority voting for classification and averaging for regression.
- Each tree is trained in a recursive manner independently.
- The goal is to find parameters of the split function which result in a good fit of the data, in order to maximize the information gain. Training stops when information gain or training set size falls below fixed thresholds.
- Decision forests prevent overfitting by training multiple de correlated trees and combining their output to achieve sufficient diversity of trees.
- The approach to calculating information gain relies on measuring similarity over the output space.

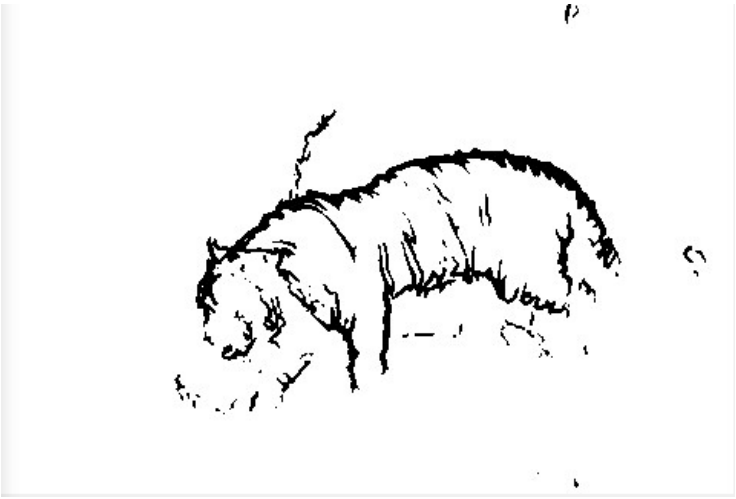
3. Results:

Probability edge map for tiger

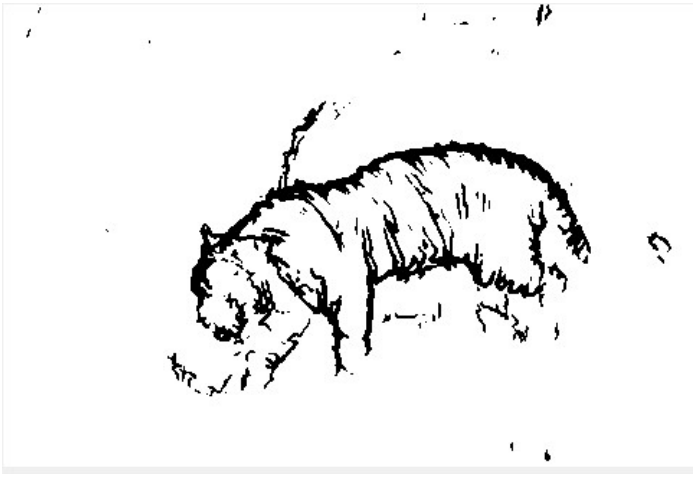
83% of classified edges included



85% of classified edges included



88% of classified edges included



90% of classified edges included

93% of classified edges included

Probability edge map with NMS

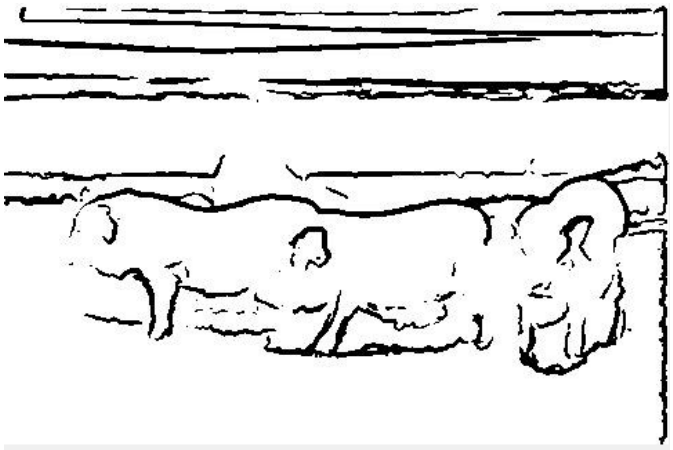
88% of classified edges included with NMS

93% of classified edges included with NMS

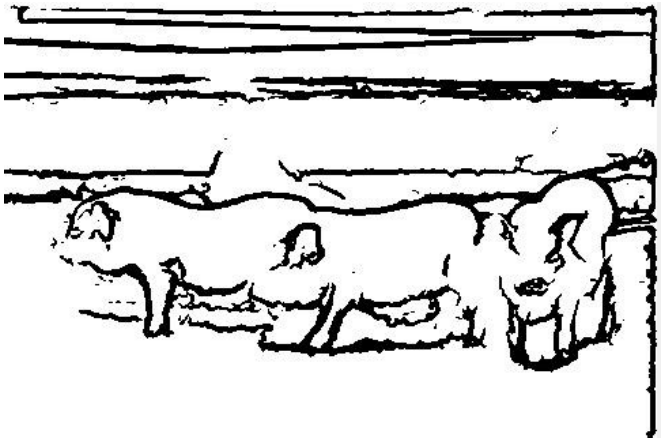
Probability edge map of pig

75% of classified edges included

80% of classified edges included

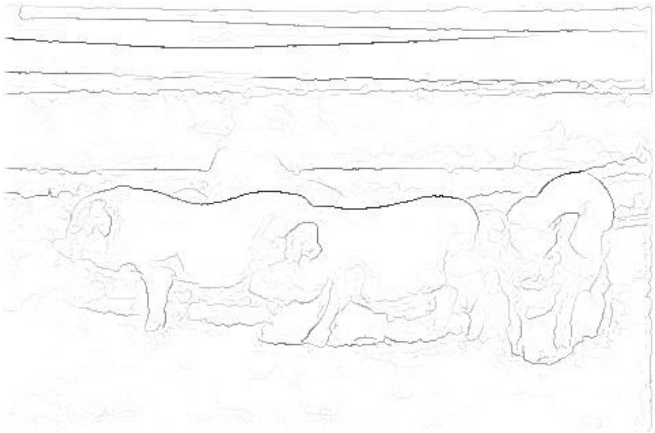


85% of classified edges included

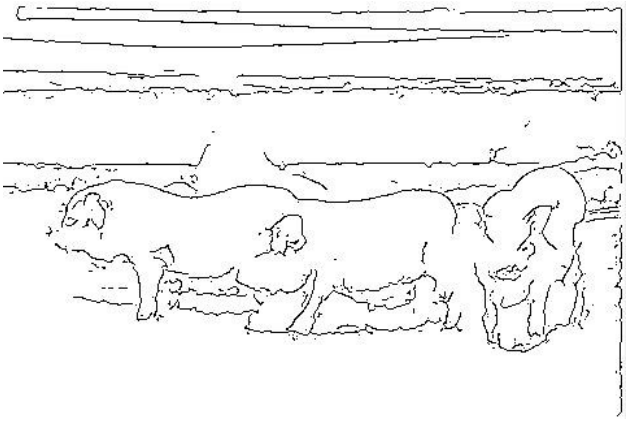


90% of classified edges included

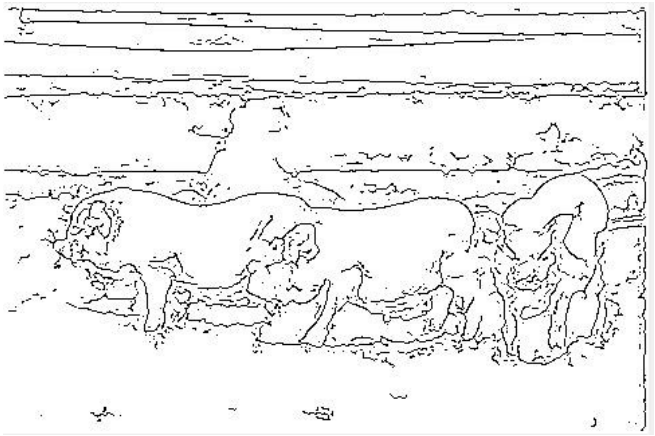
95% of classified edges included



Probability edge map, NMS included



90% of classified edges included, NMS



95% of classified edges included, NMS

4. *Discussions:*

Answer to problem 1 © 3:

Parameters:

Image patch size : 32 x 32 (computationally efficient)
Segmentation mask size: 16x 16 (optimal)
Model: Random Forest (Easy, transparent classification suited to the domain of learning from structures)
Number of threads = 4
NMS: Chosen when the percentage of classified edges is high and thinner edges are desired
No of trees taken for evaluation = 4 (since number of threads taken are 4)
Best threshold value for tiger = 0.9 (NMS = 1)
Best threshold value for pig = 0.95 (NMS = 1)

Canny Edge	Structured Edge
Differentiation based	Machine Learning Based
BASED ON VISUAL OUTPUT	
sensitive to textures	insensitive to high frequency changes which are irrelevant
does not differentiate between an edge which drops off into a background of same color	is able to differentiate between two edges of same pixel intensities
uses post processing techniques to improve differential edge detection	binarizes probability edge map, no post processing used. NMS is only used for isualization during high thresholds.

More observations:

- highly defined and continuous output
- able to identify edges and contours
- eliminates irrelevant high frequency variations (edges), insensitive to textures
- Excellent localization
- computationally efficient
- able to provide real time edge detection
- provides conclusive structure which is suitable for further application

The not-so-good-news:

- not a generative model
- need more discriminative features

PERFORMANCE EVALUATION:

To quantitatively compare the different edge
Zero Crossing Detector and SE Detector,
we use F-measure as
a parameter which is computed as follows:

$$F = 2 \text{ x } \frac{P \text{ x } R}{P+R}$$

where P stands for precision and R stands for recall. The formulas for calculating precision and recall are as follows:

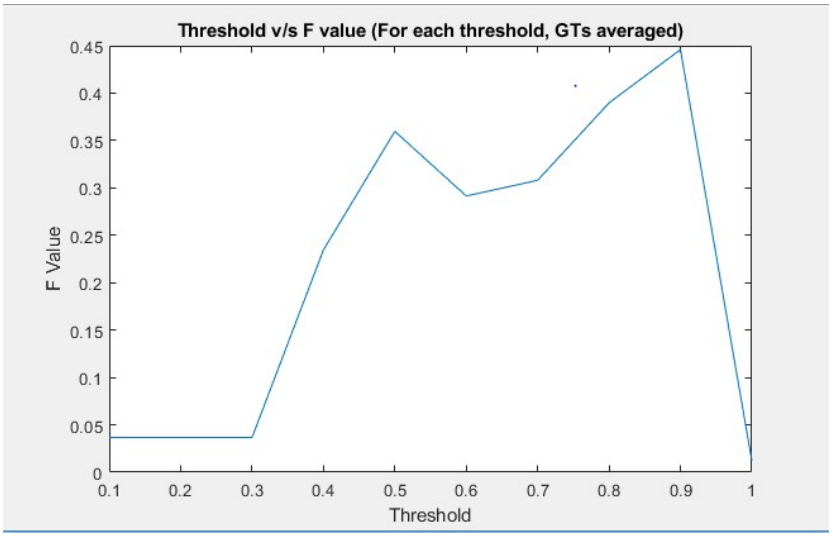
$$P = \frac{\#\ True\ Positives}{\#\ True\ Positives + \#\ False\ Positives}$$

$$R = \frac{\#\ True\ Positives}{\#\ True\ Positives + \#\ False\ Negatives}$$

Number of True Positives correspond to the number of edge pixels in the edge map that coincide with the edge pixels in the ground truth image. Hence the algorithm was correct in determining them as edges. Number of True Negatives are the number of non- edge pixels in the output edge map that coincide with the non-edge pixels in the ground truth images. These are the pixels which the algorithm was correct in not determining as edges. Number of False Positives are the number of edge pixels that the algorithms detect but aren't present in the ground truths. Lastly, the number of False negatives are the non-edge pixels in the edge map which is predicted but are supposed to be detected as edge-pixels. The groundtruth images are obtained from the BSD500 dataset which consists of human labelled and then averaged prediction maps.

Number of True Positives correspond to the number of edge pixels in the edge map that coincide with the edge pixels in the ground truth image. Hence the algorithm was correct in determining them as edges. Number of True Negatives are the number of non- edge pixels in the output edge map that coincide with the non-edge pixels in the ground truth images. These are the pixels which the algorithm was correct in not determining as edges. Number of False Positives are the number of edge pixels that the algorithms detect but aren't present in the ground truths. Lastly, the number of False negatives are the non-edge pixels in the edge map which is predicted but are supposed to be detected as edge-pixels. The groundtruth images are obtained from the BSD500 dataset which consists of human labelled and then averaged prediction maps.

This analysis was applied to the SE detection output of tiger and the following was obtained for each threshold (averaged ground truths):



Best F Value for this experiment was 0.45 at threshold of 0.9 for the tiger image.

PROBLEM B: HALFTONING

Dithering

(a)

1. **Abstract and Motivation:**
   Halftones are a technique that allows one density of ink (black) to print different tones (grays). Whether by silk screen or printing press. Look at a printed picture (newspaper or magazine) with a strong magnifier. You will see a series of solid dots of various sizes surrounded by areas of white (blank paper). This tricks your eye to see various shades. It uses the low pass filter property of human vision. Dithering explores some basic algorithms to binarize / halftone images.

2. **Approach and Procedures:**

   1. Random Thresholding: Halftone the image using randomly generated thresholds from the uniform distribution

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

   2. Dithering: Use dithering matrices to threshold input images. These dithering matrices are generated by the following algorithms:

I2 = [1 2;3 0]

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

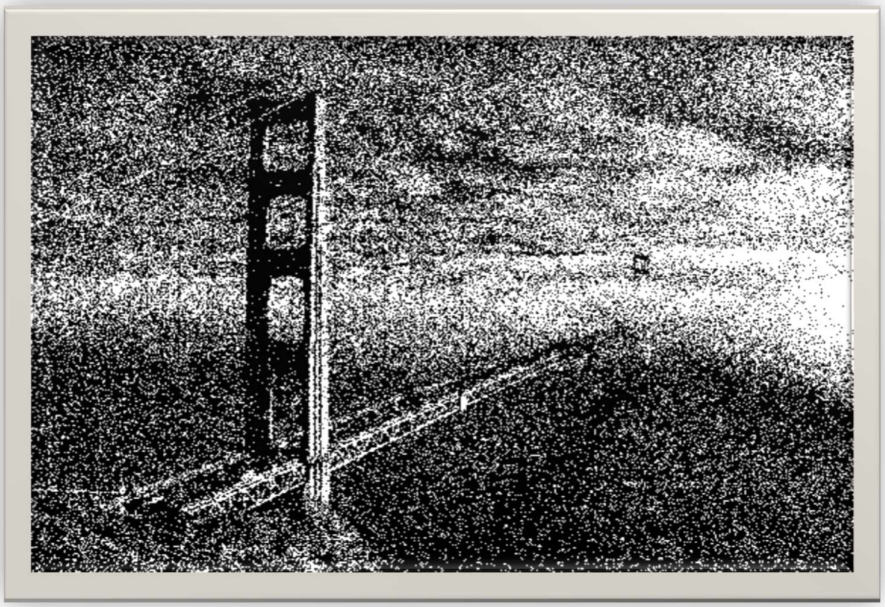
They are then thresholded and halftoned like so:

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

where N squared denotes the number of pixels in the matrix

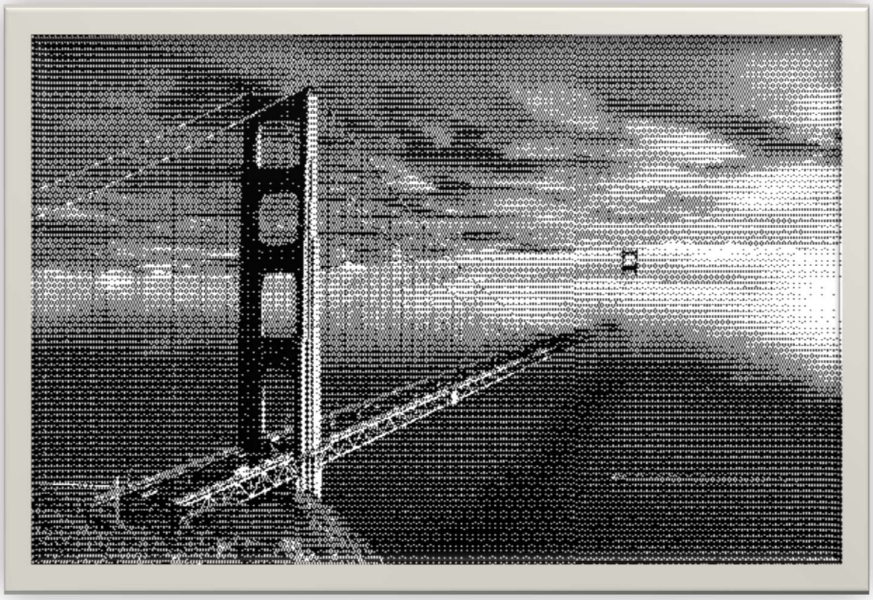
$$G(i,j) = \begin{cases} 0 & if\ 0 \le F(i,j) \le T(i\ mod N, j\ mod N) \\ 255 & T(i\ mod N, j\ mod N) < F(i,j) < 256 \end{cases}$$

3. **Results**:

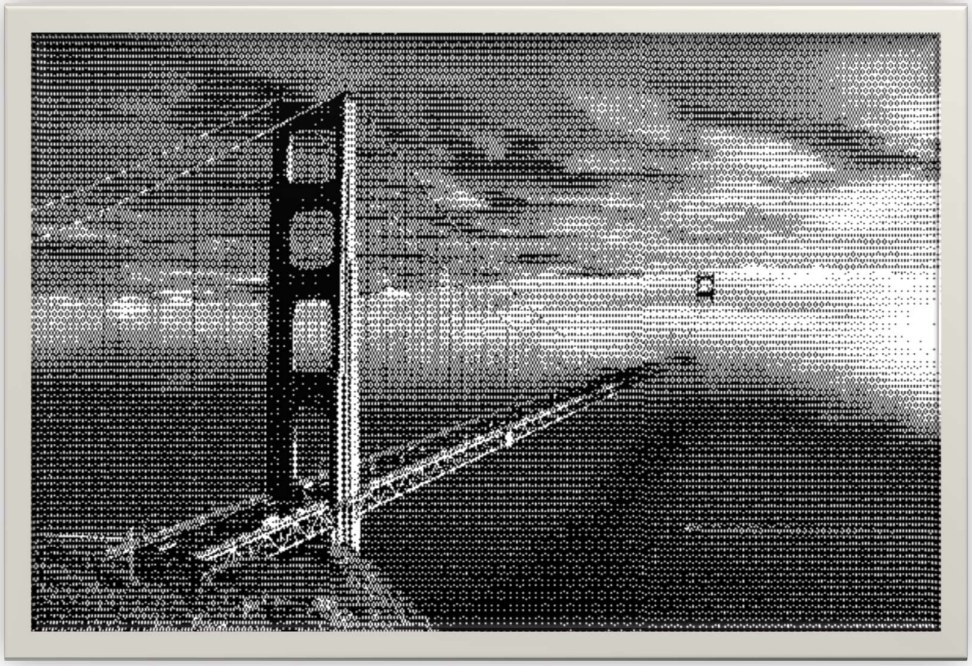


Random Thresholding

Dithering with I2 matrix



Dithering with I8 matrix



Dithering with I32 matrix

4. Discussion:

Random Thresholding	**I2 matrix**	**I8 matrix**	**I32 matrix**
Output unclear	Output clear but not well defined	Good halftoning involved	Good visual result
Random noise seen	Halftoning done in small non uniform block like patterns	When observed from afar, the aesthetic is convincing. From up close, uniform block like pattern is observed	When observed from afar, the aesthetic is convincing. From up close, uniform block like pattern is observed
Uneven halftoning	Textures of black and white are observed in patches	Smaller faithful patches observed with respect to original image	There is not much of a difference between I8 and I32 since the low pass filter capability of the eye is not able to differentiate such small variations.

Error Diffusion

(b) and (c)

1. **Abstract and Motivation:**
   The previous algorithm had a threshold that did not consider sufficiently the errors involved while the neighbourhood pixels were being halftoned. Error diffusion ensures that the error from halftoning on pixel gets diffused all the way from the top left corner through the image to the bottom right corner via serpentine scanning.

2. **Approach and Procedures:**

   All error diffusion algorithm involved except MVBQ follow the algorithm below:

   1. Convolve input image with windows of predefined error diffusion matrices (Floyd, JJN, Stucki)

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

Floyd Steinberg

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

JJN

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

Stucki

   2. Fixed threshold and calculate the error: old value – new value assigned
   3. Diffuse the error through serpentine scanning
   4. Iterate for all pixels

   Additional Setup:
   1. *Boundary extension*
   2. *Serpentine Scanning*

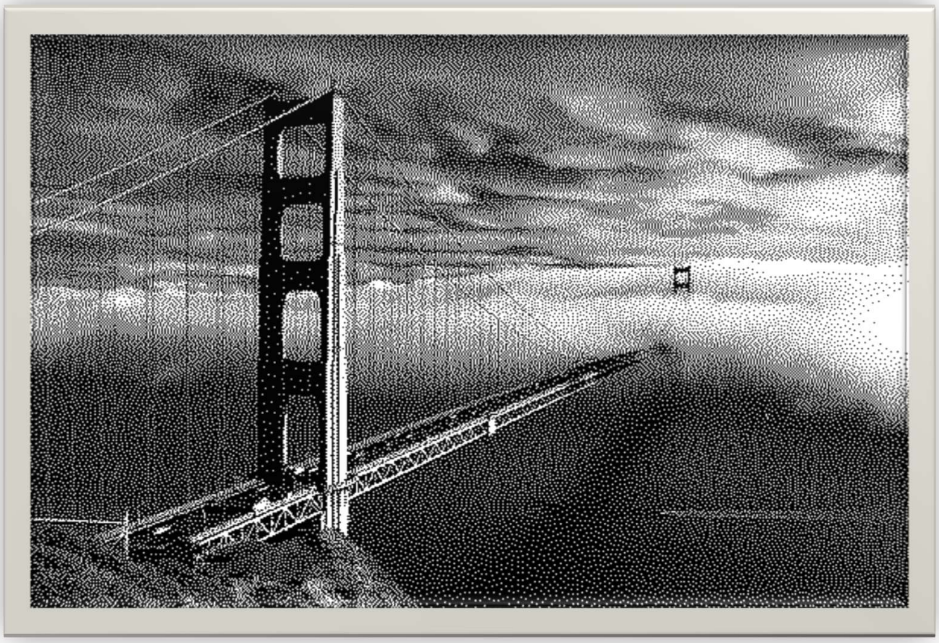
MBVQ:

Answer to 2 (1):

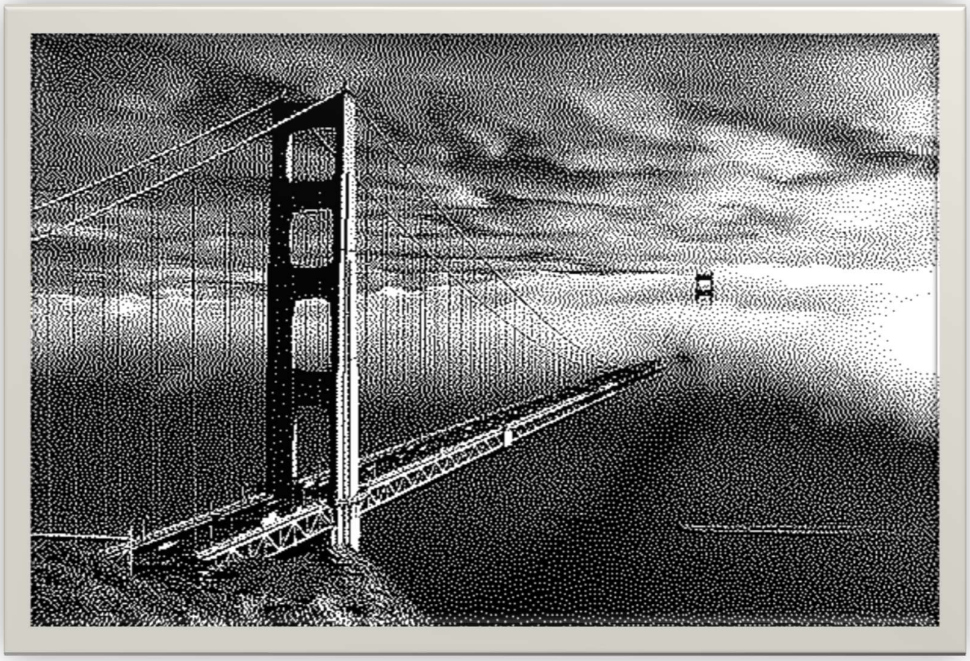
MVBQ takes into account that human visual perceptibility is more biased to brightness than chrominance. The following algorithm is followed:

1. Iterate for every pixel
2. Find the quadrant to which it belongs i.e. pertains to minimum brightness variation
3. Find the nearest pixel in the quadrant which approximates the input pixel
4. Calculate error = old value – new value
5. Diffuse the error via serpentine scanning

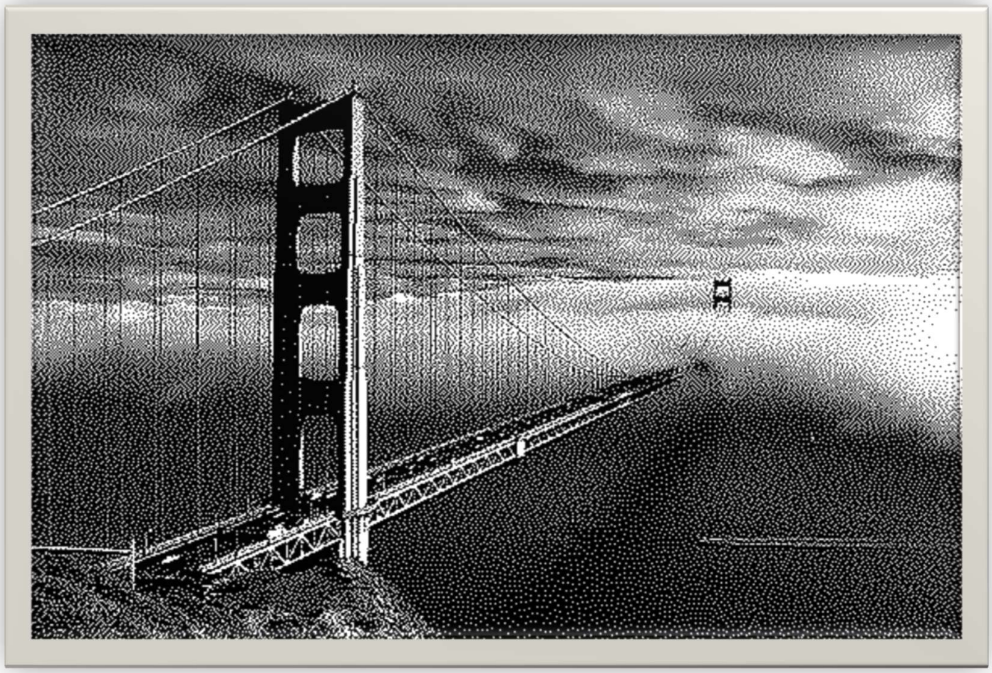
3. Results:



Floyd Steinberg Output

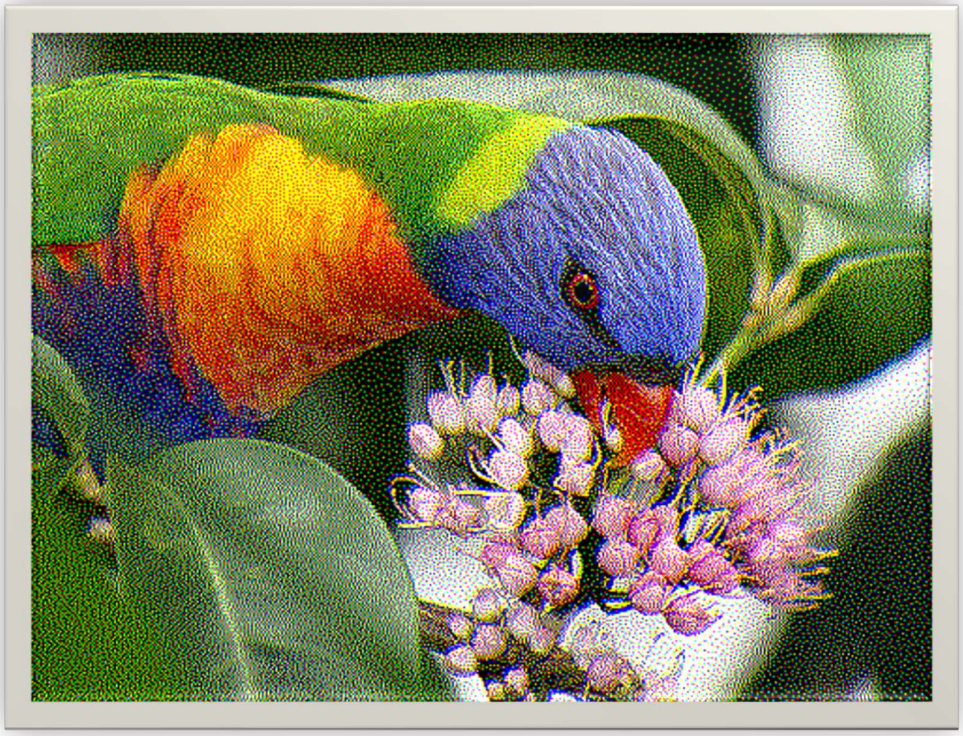


Post JJN Dithering



Post Stucki Dithering

Separable Error Diffusion

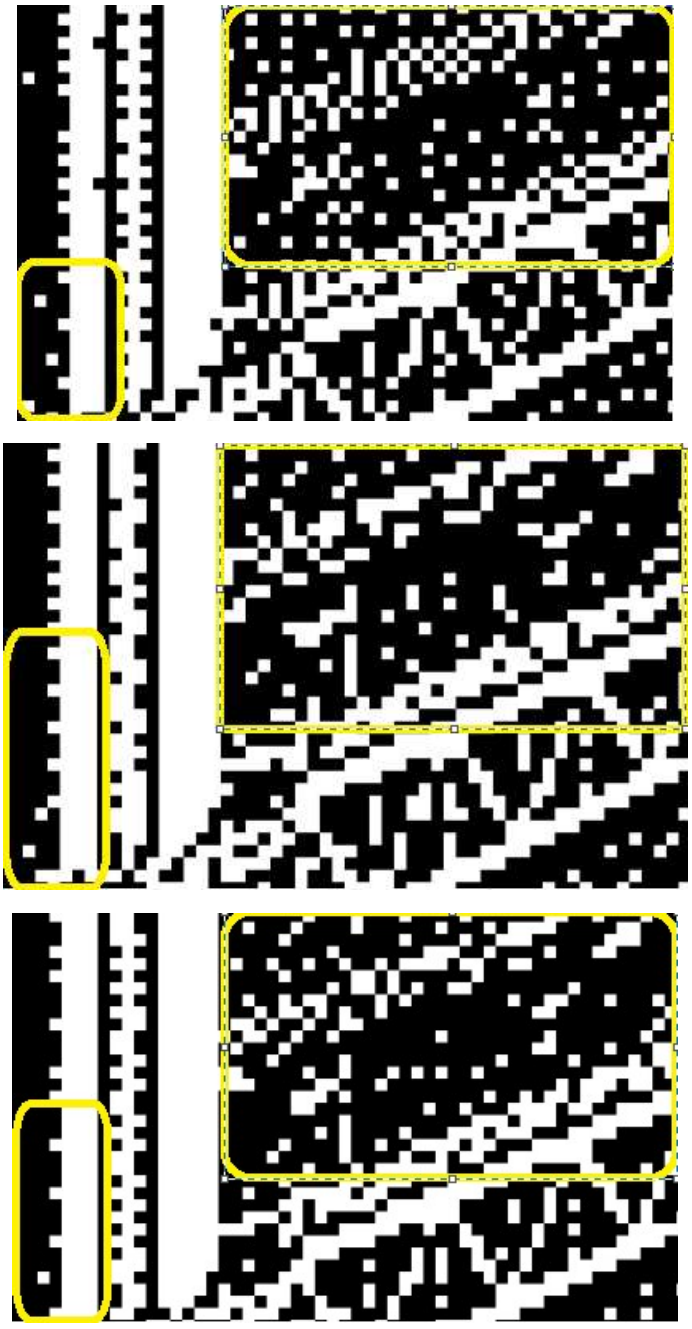


MBVQ (above)

4.Discussion:

Below are the answers to questions 1 and 2 from c in problem 2.

1. The error diffusion matrices are better than the dithering matrices as the local error is considered and variation in assignment of halftones is facilitated through its diffusion. A finer output is obtained which bypasses the low pass filter of the eye better than the dithering matrices.
2. Comparison of outputs:



In the first image above, Floyd Steinberg is used.

- Incorrect halftoning is observed in completely dark or bright areas (spots of white in black and vice versa)
- The bridge sections shows that the bridge undergoes incorrect assignment of clumps of white pixels
- Lower contrast
- Less details

In second image, JJN is used

- There is an improvement in completely dark or bright areas as shown
- Higher contrast comparatively
- More detailing
- Sharper and clearer output

In third image, Stucki is used

- Completely bright or dark areas has severe improvement
- Well defined edges and excellent detailing
- Better than Floyd Steinberg

The best algorithm of the three would have to be Stucki. As we can see from the bridge, the halftoning has resulted in a relatively uniform pattern describing the bridge and continuity and transparency of assigned pixels is poignant.

A better algorithm in my opinion would be to include

- larger window size to accommodate more variations as per neighbourhood. (area based algorithms)
- If Stucki's error matrix is improved upon, it can be accomplished by simpler computations such as bit shifting.
- Dithering and error diffusion can be combined. Error diffusion can be used to diffuse the error and dither matrices can be used as a threshold.

Color Halftoning:

Answer to parts of question 1 and 2 1)

What is the drawback of separable error diffusion and how does MVBQ overcome it?

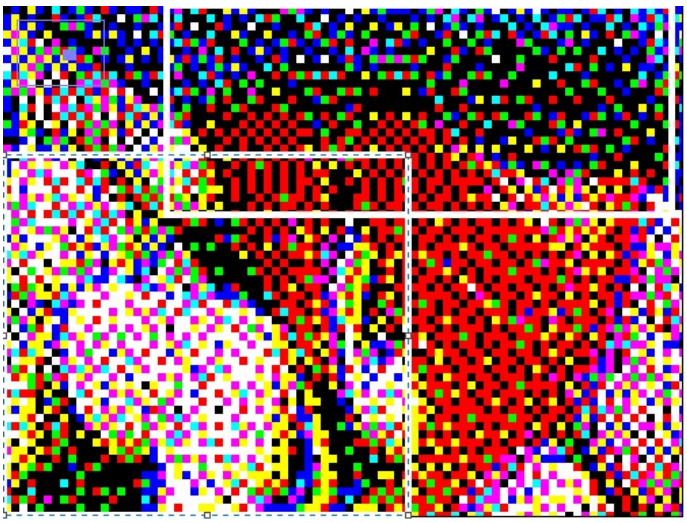
To produce a good color halftone, the following conditions must be satisfied:

- visual unnoticeability
- local color is the desired color
- the colors used reduce the noticeability of the pattern

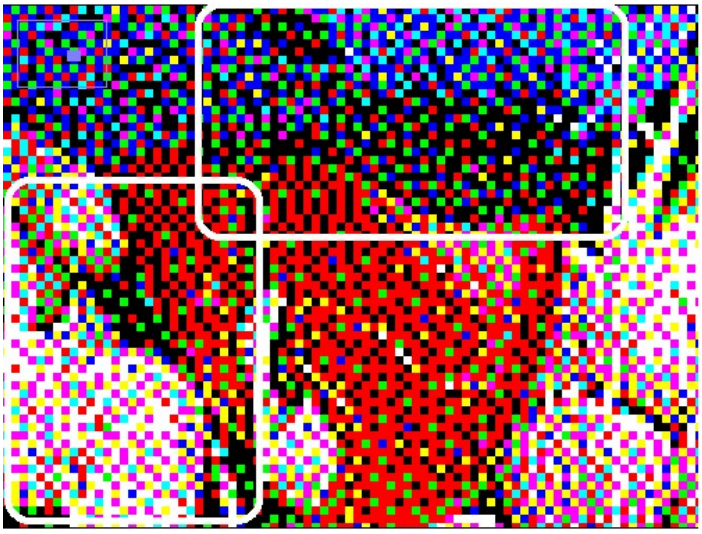
Normal diffusion algorithms take care of the first two parts but not the third part. MBVC includes *ink relocation,* which balances the tradeoff between compliance the third criteria. It takes into account the perceptibility of human vision which separable error diffusion algorithm does not. It overcomes this by considering that human vision is more susceptible to brightness than chrominance. Thus, instead of just diffusing the error, it also selects the

pixel color from the eight, the one whose brightness variation is minimal. Moreover, Separable error diffusion does not exploit the inter color correlation, hence it leads to color artifacts and poor color rendition.

- **Comparison (Visual):**



Separable Error Diffusion (with white highlighting box)



MBVQ (with white highlighting box)

- Here, we can see that in the top box, there are more black pixels using the separable ED method. These are better quantized in MBVQ.
- There are artefacts in the bottom left section of Separable ED due to lack of inter color correlation. These artefacts are improved upon in MBVQ.

- MBVQ looks for closest vertex in quadrant of minimum variation whereas the latter looks for the closest vertex in the cube.
- MBVQ is preferred over separable error diffusion as it reduces halftone noise as observed
- Regarding run time, MBVQ takes little more time than separable error diffusion as MBVQ has high pixel density, retains local averaging and reduces noise which results in more time.