Homework Two Report

Zifan Wang

February 13, 2019

1 Problem 1: Edge Detection

1.1 A. Sobel Edge Detector

1.1.1 Abstract and Motivation

Humans classify the object based on its contours and textures. In order to help computer recognize the contours or the shape of an object, people develop a edge detection method. **Sobel Edge Detection** method is one of the classic edge detection method.

1.1.2 Approach and Procedures

First, convert RGB input image to the gray-scale image by following equation:

$$G(i,j) = 0.2989 \times I(i,j,0) + 0.5870 \times I(i,j,1) + 0.1140 \times I(i,j,2)$$

where the $I(i,j,0)$ is red color value at the input image position (i,j), the $I(i,j,1)$ is green color value at the input image position (i,j), and the $I(i,j,2)$ is blue color value at the input image position (i,j).
Second, use a 3×3 G_x and G_y to calculate the derivative of x direction and y direction for each pixel in the image:

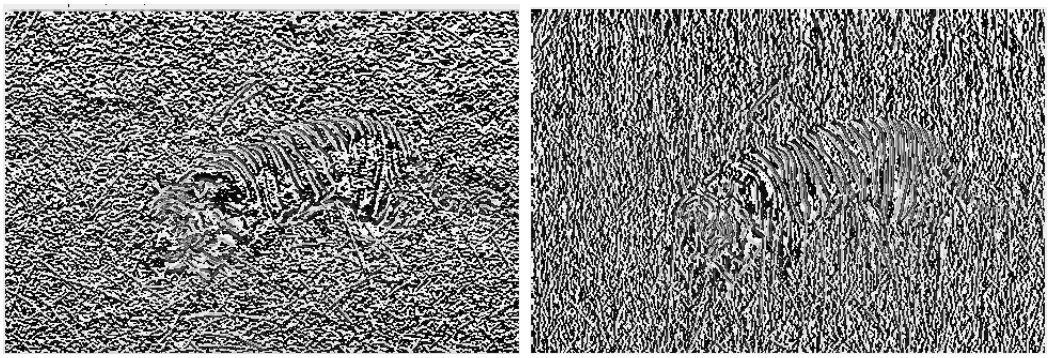
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Third, calculate the gradient magnitude by $\sqrt{\frac{\partial F}{\partial x}^2 + \frac{\partial F}{\partial y}^2}$. Last, apply threshold to display the final edge map.

1.1.3 Result

The figure 1 shows the intermediate step of calculating the final output. These two images comes from after applying G_x and G_y filters.



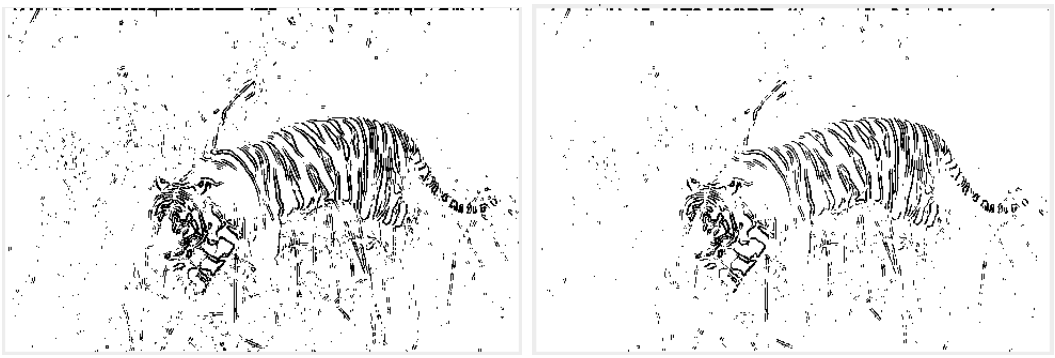
(a) Horizontal Edge (b) Vertical Edge

Figure 1: X and Y Direction Derivative of Tiger Image

(a) Tiger Egde Normalized (b) Pig Egde Normalized

Figure 2: Normalized Edge Map

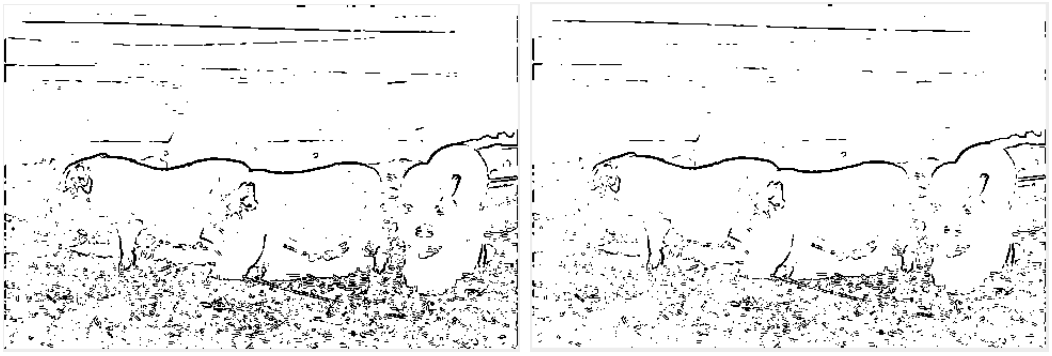
The figure 2 shows the normalized gradient edge maps of pig and tiger.



(a) Tiger with threshold 92.5% (b) Tiger with threshold 95%

Figure 3: Sobel Edge Map of Tiger Image

The figure 3 shows tiger's edge with two different threshold percentages.



(a) Pig with threshold 92.5% (b) Pig with threshold 95%

Figure 4: Sobel Edge Map of Pig Image

The figure 4 shows pigs' edges with two different threshold percentages.

1.1.4 Discussion

Comparing the figure 3(a) with the figure 3(b), the high percent threshold will remove many texture point and gives a better result. However, it will also remove some true edges which are not very clear to capture.

1.2 B. Canny Edge Detector

1.2.1 Abstract and Motivation

People are not satisfied with the edge map calculated by Sobel Edge detector. John F. Canny developed a multi-stage algorithm to detect a wide range of edges in images in 1986. This new method called **Canny Edge Detection** performs better than Sobel Edge detector.

1.2.2 Approach and Procedures

Canny edge detection algorithm can be separate into 5 steps:

- Change the input image from RGB color space to gray-scale.
- Apply the Gaussian filter to remove the noise in the gray-scale image.
- Find the gradients of the image.
- Use non-maximum suppression to get rid of spurious response in edge detection.
- Apply double threshold to determine the potential edges.
- Suppress all weak and not connected edges.
- Output edge map.

For converting RGB input image to the gray-scale image, the following equation is applied:

$$G(i,j) = 0.2989 \times I(i,j,0) + 0.5870 \times I(i,j,1) + 0.1140 \times I(i,j,2)$$

where the $I(i,j,0)$ is red color value at the input image position (i,j), the $I(i,j,1)$ is green color value at the input image position (i,j), and the $I(i,j,2)$ is blue color value at the input image position (i,j).

Gaussian filter: The following equations are used to generate the Gaussian filter and output image:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(k-i)^2 + (l-j)^2}{2\sigma^2}\right)$$

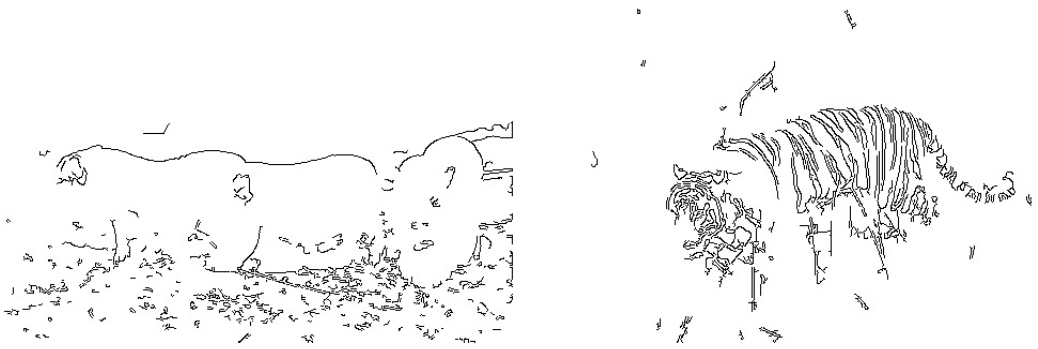
Find Gradient: Apply Sobel Edge Detection's gradient finding method to the gradient of each pixel.

Non-Maximum Suppression: Non-maximum suppression is used to find the largest edge. This method will suppress all the gradient values by setting them to 0 and only keep the local maxima which indicates the sharpest change of intensity value. The non-maximum suppression method is implemented by comparing the edge strength of the current pixel to the edge strength of the pixel in the positive and negative gradient directions. If the current pixel has the largest edge strength, the value will be keeped. Otherwise, this strength value will be suppressed.

Double Thresholding: To further remove spurious edge, the double thresholding method is applied to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient values. The double thresholding contains a low threshold value and a high threshold value. If an edge pixel's gradient is higher than the high threshold, it will be marked as a strong edge. If a pixel gradient value is lower than the small threshold, it will be set as background. If the pixel gradient is between the two thresholds, it will be marked a weak edge.

Edge Further Classification: This method will reduce weak edges which do not connect to the strong edge.

1.2.3 Result



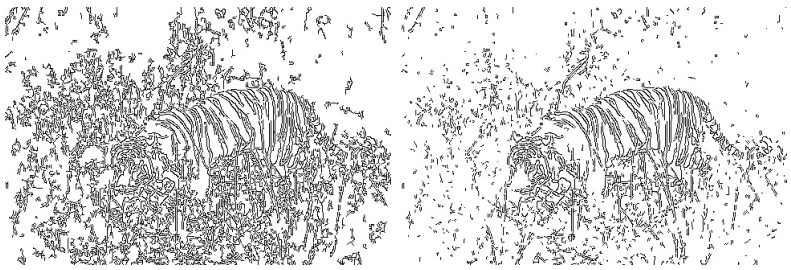
(a) Pig Edge Map (b) Tiger Edge Map

Figure 5: Edge Map Generated by Canny Edge Detector

The figure 5 shows the pig's and tiger's edge maps generated by Canny Edge Detection method. For pigs' edges detection, the low threshold is set to 200 and the high threshold is set to 400. For tiger's edge detection, the low threshold is 200 and the high threshold used is 600.

1.2.4 Discussion

Double threshold method is important for edge detection in Canny Edge Detection method. The low and high thresholds will set true edge and remove the unnecessary edge. As a result, it is necessary to experiment with these thresholds. Keep high threshold as constant to notice the effect of low threshold and keep the low threshold as constant to see the effect of high threshold.



(a) Low Threshold = 100 (b) Low Threshold = 200

Figure 6: Low Threshold Value Comparison

The figures in 6 have the same high threshold value which equals to 300. Comparing the figure 6(a) with the figure 6(b), the higher low threshold value can remove more unnecessary information which is not edge from the image. However, the large low threshold value can also remove the edge from the image. The figure 7 below shows this effect:

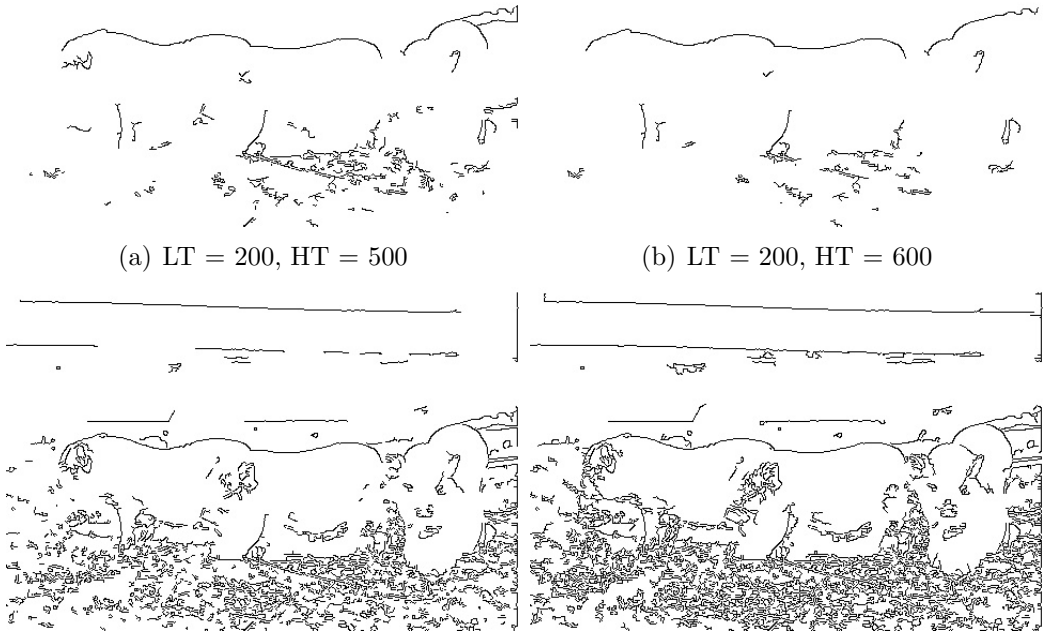
Figure 7: Low Threshold = 450, High Threshold = 600

The figures in 8 have the same low threshold value which equals to 450. Comparing the figure 8(a) with the figure 8(b), the higher the high threshold value can also remove some unnecessary information which is not edge from the image. However, it further reduces some true edges in the tiger.

(a) High Threshold = 500 (b) High Threshold = 700

Figure 8: High Threshold Value Comparison

The follow figure 9 shows the experimental result of pig image. The conclusion before does match the output of pig's experiment.



(a) LT = 200, HT = 500 (b) LT = 200, HT = 600

(c) LT = 150, HT = 300 (d) LT = 100, HT = 300

Figure 9: Pig Experiment Result

Although the Canny Edge Detection method does not perform too well, it still better than the Sobel Edge Detection method by comparing the figure 3 with the figure 5.

1.3 C. Structured Edge

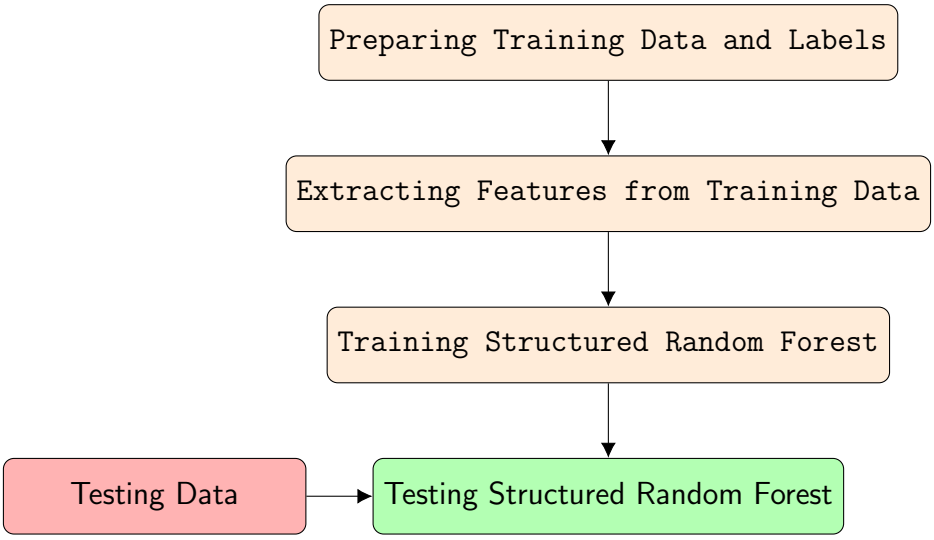
1.3.1 Abstract and Motivation

With the help of computers' development, the high speed calculation support people to develop data driven method. One of the famous data driven method used in edge Detection is **Structured Edge**. Different from the Sobel and Canny Edge detector, the Structured Edge detection algorithm applies decision tree and edges' templates to detect edge. This technique will detect the edge more clearly with less wrong labeling.

1.3.2 Approach and Procedures

Structured Edge Algorithm:
The Structured Edge algorithm involves training random forest part and testing this forest part. Random decision forest trained by images' patches with correct labels: edge or no edge gives patches of testing images labels: edge or no edge. By using these labels, a final edge map can be reconstructed. The performance evaluation can be calculated to decide whether this structured random forest is good enough or not. The flow chart

below shows the processes of structured edge algorithm:



- First, crop training images into small patches and label each patches edge or no edge.
- Second, augmenting each image patch with multiple additional channels' information will result a feature vector $x \in \mathbb{R}^{n \times n \times k}$ where n is the size of image patches and k is the number of color channels. To build a decision tree, pixel lookups: $x(i, j, k)$ and pairwise differences $x(i_1, j_1, k) - x(i_2, j_2, k)$ are used. In addition, gradients and color is computed for each patches.
- Third, the random forest is formed by decision trees and the decision tree is built by internal leaf nodes. These leaf nodes are generated by the extracted features from the previous step and decision thresholds. Passing the training data to the decision tree, the output space Y will be generated. To map all the structured labels y (*edge or no edge label from previous step*) $\in Y$ at a given node into a discrete set of labels $c \in C$, where $C = \{1, ..., k\}$, the intermediate mapping and information grain criterion are developed. Intermediate mapping: $\Pi : Y \to Z$ is approximate dissimilarity of $y \in Y$ by computing Euclidean distance in Z which is calculated by pairwise differences from the previous part. Sample m dimensions of Z, resulting in a reduced mapping Π_θ parametrized by θ. In the training, a distinct mapping Π_θ is randomly created. This can result a fast computing, can ensure sufficient diversity, and can reduce dimensions. Information grain criterion is the last step to from the output of one lead node. The Principal Component Analysis (PCA) is used to reduce the dimensionality of Z and form an most significant space for z. In this z, using the k-means method (k = 2) to divide the samples entering the leaf node into two classes. Then, we can decide whether the feature selected for the internal leaf node is available or not and find the optimal threshold for this splitting. Iterate the processes of building one leaf node to build a decision tree. Last, generate a random forest by building numbers of different decision trees.
- Last, use the testing images to evaluate the performance of built random decision forest.

Decision Tree Construction

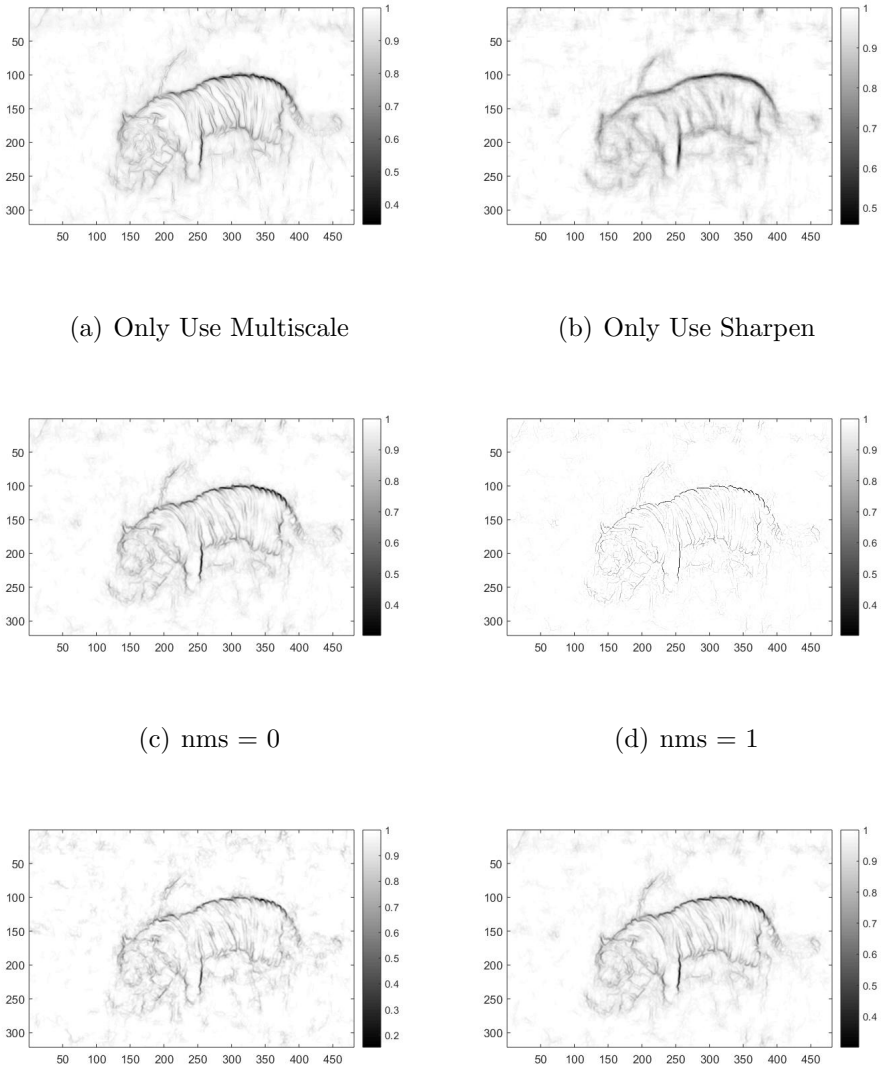
The decision tree is a tree structure and has the internal leaf nodes and final leaf node.

The internal leaf node contains a threshold and feature to classify the input samples into the following nodes. Iterating through the internal leaf nodes, the input samples will reach the final leaf node. This type of leaf nodes set a label to the input sample reaching it. Internal leaf nodes' optimal features and thresholds are important in this building process. In addition, the decision tree should be shallow to prevent over-fitting.

Principle of RF classifier

The random forest classifier uses numbers of decision trees to classify samples. Integrating the results of these decision trees into one final probability function, the random forest classifier has better classification rate and prevent over-fitting problem in the decision tree.

1.3.3 Results



(a) Only Use Multiscale

(b) Only Use Sharpen

(c) nms = 0

(d) nms = 1

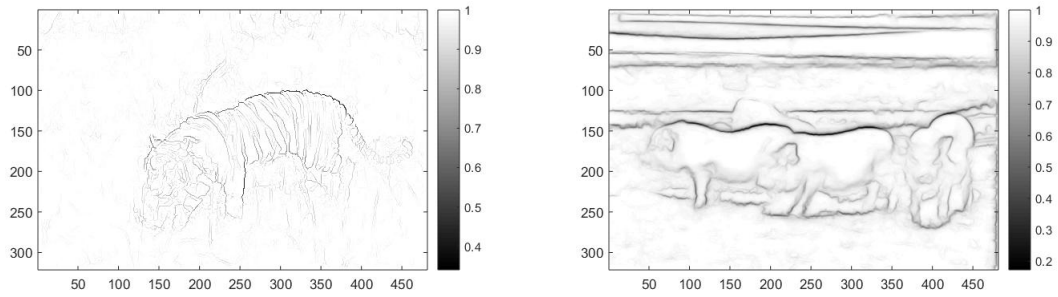
(e) nTreesEval = 1

(f) nThreads = 4

Figure 10: Experiment with Detection Parameters

The figure 10 experiment with edge detection parameters in the given Structed Edge

Detection codes. Comparing the figure 10(c) with the figure 10(d), the mean-square turns on will produce a better result. The sharpen is used to sharpen the edge by some scale. nTreesEval is the parameter used for set number of decision trees in the RF classifier and nThreads is number of threads. After experiment, I set model.opts.multiscale = 1, model.opts.sharpen = 3, model.opts.nTreesEval = 5, model.opts.nThreads = 4, and model.opts.nms = 1. The following figure 11 is my result:



(a) Tiger (b) Tiger

Figure 11: My result

1.3.4 Discussion

In the figure 3and the figure 4, the **Sobel Edge Detector** generates more redundant edge points. In addition, some edge points are not continuous as they suppose to be. The figure 5 clearly shows that **Canny Edge Detector** has some improvements comparing to **Sobel Edge Detector**. However, it loses some edge information. From the figure 11, the **Structured Edge Detector** overcomes the drawbacks of **Sobel Edge Detector** and **Canny Edge Detector**. It suppresses unnecessary edges without removing the important ones.

1.4 Performance Evaluation

1.4.1 Abstract and Motivation

Although **Structured Edge Detection** is a powerful tool for computer to detect edges, the miss labeling can happen during the processes. People develop a performance evaluation to evaluate the performance of each edge detection method by comparing the edge map generated by edge detection methods with human labeling edge map (called the ground truth).

1.4.2 Approach and Procedures

Before performance edge evaluation, the following four classes can be define:

- True positive: Edge pixels in the edge map coincide with edge pixels in the ground truth. These are edge pixels the algorithm successfully identifies.

- True negative: Non-edge pixels in the edge map coincide with non-edge pixels in the ground truth. These are non-edge pixels the algorithm successfully identifies.
- False positive: Edge pixels in the edge map correspond to the non-edge pixels in the ground truth. These are fake edge pixels the algorithm wrongly identifies.
- False negative: Non-edge pixels in the edge map correspond to the true edge pixels in the ground truth. These are edge pixels the algorithm misses.

The precision, P is calculated by $\frac{\#True\ Positive}{\#True\ Positive + \#False\ Positive}$.
The recall, R is calculated by $\frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$.
F is defined by $2\frac{PR}{P+R}$
In addition to handle humans' opinion diversity, the mean precision and mean recall is used. Based on the mean precision and mean recall to calculate the F.

1.4.3 Result

The following tables show the result of performance evaluation of each edge detection method.

Tiger	Precision	Recall	F
groundTruth 1	0.0563	0.9450	
groundTruth 2	0.0602	0.9525	
groundTruth 3	0.0672	0.9477	
groundTruth 4	0.2762	0.9873	
groundTruth 5	0.0705	0.8567	
Mean	0.1061	0.9378	0.1906

Table 1: Tiger Image Performance Evaluation of Sobel Edge Detector

Pig	Precision	Recall	F
groundTruth 1	0.0740	0.6704	
groundTruth 2	0.0793	0.6647	
groundTruth 3	0.1054	0.5514	
groundTruth 4	0.1363	0.5673	
groundTruth 5	0.1026	0.5496	
Mean	0.0995	0.6007	0.1707

Table 2: Pig Image Performance Evaluation of Sobel Edge Detector

Tiger	Precision	Recall	F
groundTruth 1	0.0681	0.9450	
groundTruth 2	0.0731	0.9568	
groundTruth 3	0.0817	0.9523	
groundTruth 4	0.3321	0.9813	
groundTruth 5	0.0757	0.7598	
Mean	0.1261	0.9196	0.2218

Table 3: Tiger Image Performance Evaluation of Canny Edge Detector

Pig	Precision	Recall	F
groundTruth 1	0.0981	0.8154	
groundTruth 2	0.1055	0.8105	
groundTruth 3	0.1095	0.5254	
groundTruth 4	0.1129	0.4311	
groundTruth 5	0.1013	0.4979	
Mean	0.1055	0.6160	0.1801

Table 4: Pig Image Performance Evaluation of Canny Edge Detector

Tiger	Precision	Recall	F
groundTruth 1	0.533	0.625	
groundTruth 2	0.616	0.732	
groundTruth 3	0.605	0.503	
groundTruth 4	0.546	0.817	
groundTruth 5	0.325	0.778	
Mean	0.525	0.691	0.597

Table 5: Tiger Image Performance Evaluation of Structured Edge Detector

Pig	Precision	Recall	F
groundTruth 1	0.478	0.561	
groundTruth 2	0.453	0.595	
groundTruth 3	0.242	0.503	
groundTruth 4	0.459	0.617	
groundTruth 5	0.393	0.678	
Mean	0.405	0.591	0.480

Table 6: Pig Image Performance Evaluation of Structured Edge Detector

1.4.4 Discussion

The tables above shows that Structured Edge Detector has the best performance in detecting edge. The next is the Canny Edge Detector. And the last is Sobel Edge detector. Only using Gx and Gy to calculate the gradient magnitude, the Sobel edge detection method generates lots of false positive edges. Therefore, its performance is worse comparing to Canny and Structured Edge Detector. Applying non-maximum suppression and double thresholding algorithms, the Canny Edge detector the precision is much improved compared to the Sobel Edge detection. In other words, the Canny Edge detector suppress

some unnecessary edges (true negative edges) to improve this accuracy. The Structured Edge detector adopts data driven method to reduce the number of false positive and false negative edges. As a result, it has the best performance in those three detection methods.

Tiger image is easier to get a high F measurements. The original tiger image only contains tiger and the grass. There will be a significant jumping in the edge points. This feature is easy to detect by the computers. However, the original pig image shows that the head parts of pigs are hard for computers to recognize. The reason is that the heads' color is same to the body color of pigs. Therefore, the edge detection methods will hard to detect the edge there. As a result, tiger image will intuitively have a high F measurement.

Recall that the prediction results are in the four categories: true positive, true negative, false positive, and false negative. People cannot use only one category to judge whether the edge detection is good or bad. Therefore, the F measurement combines these four categories to decide the edge detection performance. The high precision will result a low recall.

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

From the equation above, we can notice that when P is large and R is small, the F will be approached to zero. Similarly, when R is high and P is low, the F will also approached to zero. When the sum of P and R is a constant, $C = P + R$. Therefore, $P = C - R$. We can use the equation above to get the highest F value:

$$F = 2 \cdot \frac{R \cdot (C - R)}{R + C - R}$$

$$F = 2 \cdot \frac{R \cdot (C - R)}{C}$$

The highest F will be gotten, when $R = \frac{C}{2}$. Therefore, $R = P = \frac{C}{2}$

2 Problem 2: Digit Half-toning

2.1 Dithering

2.1.1 Abstract and Motivation

Halftone is a technique that simulates continuous tone imagery by using dots, varying either in the size or in spacing. Therefore, it will generate a gradient-like effect. Printers and the publishing industry widely adopt this technique, because most printers are not able to reproduce shifting of gray intensity. Digital half-toning is similar to half-toning in which an image is decomposed into a grid of halftone cells. Objects of an image are simulated by filling the appropriate halftone cells. The more number of black dots in the halftone cell, the darker the cell appears. There are different kinds of methods to determine which dot is going to be filled. In the dithering section, **Random Thresholding** and **Dithering Matrix** methods will be introduced.

2.1.2 Approach and Procedures

A gray-scale image falls in the range of 0 o 255. Compare each location's gray intensity value to the threshold value to decide whether this location should be printed as black in the output image. The following equation will present this **thresholding** idea in math formula.

$$output(i,j) = \begin{cases} 0 & 0 \leq input(i,j) < threshold \\ 255 & threshold \leq input(i,j) < 256 \end{cases}$$

where $input(i,j)$ in the pixel value at original image location (i,j) and threshold is the value used to determine whether this location should be dark or white.
The **constant thresholding** method is defined below:

$$output(i,j) = \begin{cases} 0 & 0 \leq input(i,j) < threshold \\ 255 & threshold \leq input(i,j) < 256 \end{cases}$$

where threshold is a constant value and usually assigned to 127.
The **random thresholding** method creates random threshold number at each pixel location's (i,j). The following equation defines this method:

$$output(i,j) = \begin{cases} 0 & 0 \leq input(i,j) < rand(i,j) \\ 255 & rand(i,j) \leq input(i,j) < 256 \end{cases}$$

where $rand(i,j)$ is a uniformly distributed random variable.
Dithering matrix pattern defines a pattern that indicates the pixel most likely to be turned on (to be black color). A index 2 dithering matrix is given by:

$$I_2 = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

The higher dimension dithering matrix can be calculated by the Bayer's optimum index Matrix:

$$I_{2n} = \begin{bmatrix} 4 \times I_n + 1 & 4 \times I_n + 2 \\ 4 \times I_n + 3 & 4 \times I_n \end{bmatrix}$$

The threshold matrix is defined as $T(x,y) = \frac{I(x,y)+0.5}{N^2} \times 255$. The output image will be

created by comparing the color intensity value of original image to the thresolding matrix:

$$output(i,j) = \begin{cases} 0 & 0 \leq input(i,j) < T(i\ modN, j\ modN) \\ 255 & T(i\ modN, j\ modN) \leq input(i,j) < 256 \end{cases}$$

where (i, j) is the pixel location at the original image and N is the size of the dithering matrix.
After applying **Random Thresholding** and **Dithering Matrix** methods, the output image's color will only be 0 (Black) and 255 (White). Thus, images can be easily printed.

2.1.3 Result:

Random Thresholding:
The figure 12 shows the half-tone image generated by random thresholding.

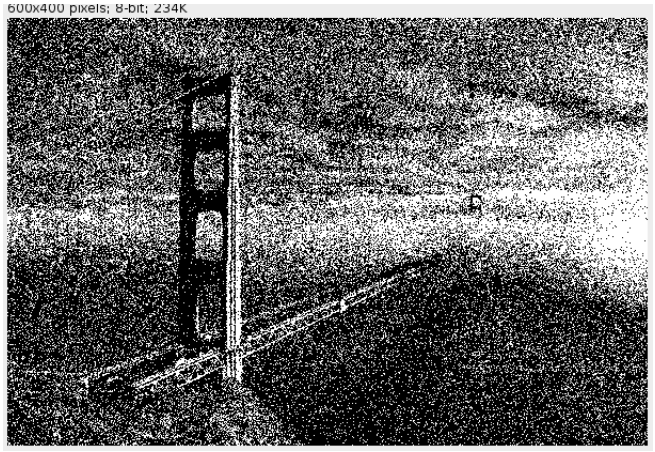


Figure 12: Image After Random Thresholding

Index 2 Dithering Matrix
The figure 13 below shows the half-tone image generated by index 2 dithering matrix.

Figure 13: Image After Index 2 Dithering Matrix

Index 8 Dithering Matrix:
The figure 14 below shows the half-tone image generated by index 8 dithering matrix.
Index 32 Dithering Matrix:
The figure 15 below shows the half-tone image generated by index 32 dithering matrix.

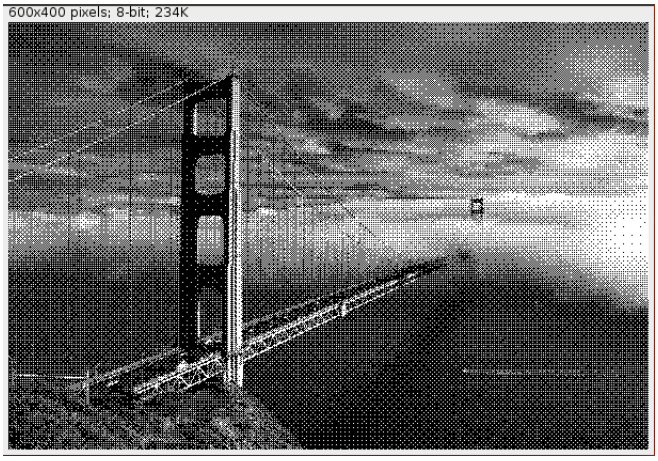


Figure 14: Image After Index 8 Dithering Matrix

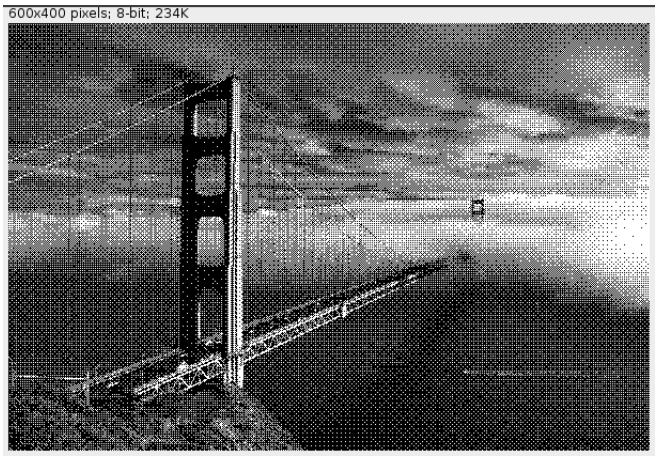


Figure 15: Image After Index 32 Dithering Matrix

2.1.4 Discussion

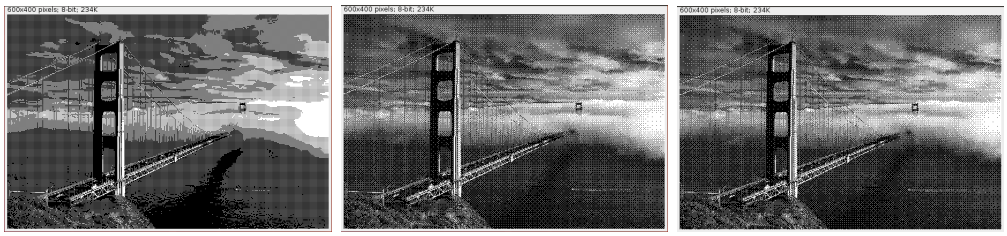
Random Thresholding:

(a) Random (b) Constant

Figure 16: Random Thresholding vs. Constant Thresholding

Comparing the figure 16(a) to the figure 16(b), the random thresholding method preserve more details than the constant thresholding method. However, the random method will introduce some random noise during the process. The PSNR value of the image after

random thresholding is 7.514.
Dithering Matrix:



(a) I_2 (b) I_8 (c) I_{32}

Figure 17: I_2 vs. I_8 vs. I_{32}

The figure 17 shows that as index increases, the half-tone image becomes better. The 17(a) has PSNR value: 7.41. The 17(b) has PSNR value: 7.53. The 17(c) has PSNR value: 7.54. In conclusion, comparing the figure 17 to figure 16(a), the higher order dithering matrix method generates better half-tone image than the random thresholding method with less noise and preserving more details.

2.2 B. Error Diffusion

2.2.1 Abstract and Motivation

To improve the half-toned images' qualities, the **Error Diffusion** method was inspired. This method quantizes each pixel using a neighborhood operation instead of a simple pointwise operation.

2.2.2 Approach and Procedures

Error Diffusion sequentially traverses each pixel of the original image. Each pixel is compared to a threshold. If the pixel value is higher than the threshold, a 255 is outputted; and vice versa. The difference between the original pixel value and the output value which is the error is dispersed to nearby neighbors. The figure below shows a loop diagram of error diffusion method. The Quantizer which shows in the figure 18 is defined

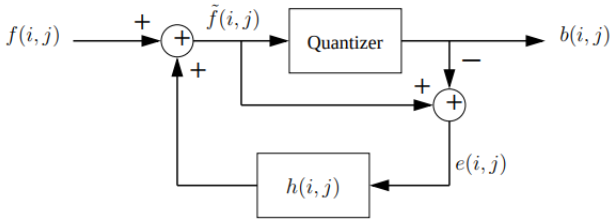


Figure 18: Error Diffusion Method Graph

as following:

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

where *threshold* is 127.
The error is defined by: $e(i,j) = \tilde{f}(i,j) - b(i,j)$. The $\tilde{f}(i,j)$, cumulative pixel value at

location (i, j) is defined by $\tilde{f}(i,j) = f(i,j) + \sum_{k,l \in S} h(k,l)e(i-k, j-l)$. There are three types of error diffusion matrices: Floyd_Steinberg, JJN, and Stucki.

- Floyd_Steinberg (by Floyd and Steinberg in 1976)

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

- JJN (by Jarvis, Judice, and Ninke in 1976)

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

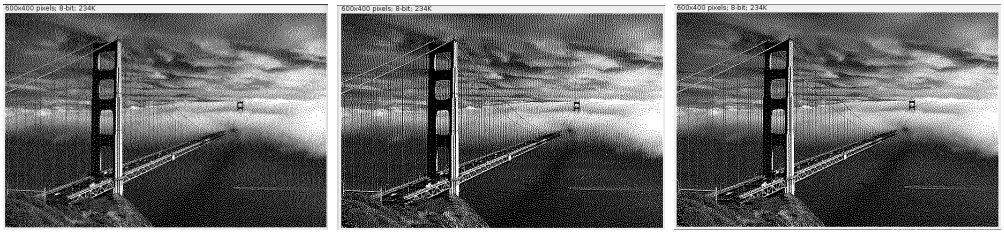
- Stucki

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

The scanning order is important to results, because when applying the filter through the image, the error will be added up. Applying raster parsing will cause the error in the right corner of filters increases. As a result, the serpentine parsing method will be used to decrease this effect.

2.2.3 Result

The following figure 18 is the result of applying error diffusion method.



(a) Steinberg (b) JJN (c) Stucki

Figure 19: Image After Error Diffusion Method

2.2.4 Discussion

The following table shows the PSNR value of output image by each gray-scale digital half-toning method: The error diffusion method has the best performance comparing to the dithering matrix method and the random thresholding method. Separating errors to the neighborhood pixels will help reconstruction a output image by persevering more

Method	PSNR
RT	7.51
I_2	7.41
I_8	7.53
I_{32}	7.54
FS	7.88
JJN	7.73
Stucki	7.63

Table 7: Table to test captions and labels

details. Since the Floyd_Steinberg's error diffusion has the highest PSNR value, it is the method I prefer. Not only based on the PSNR value, but also the smaller size filter will well reconstructed the image locally. It is clearly to see that there are lots of dots in the output image. The additive filter can be used to remove the white dots which surrounding by the dark region. In addition, the subtractive filters can be applied to remove the dark dots which surrounding by the white region. This method will further reduce the noise to produce better images.

2.3 C. Color Halftoning with Error Diffusion

2.3.1 Abstract and Motivation

Some color printers are also not be able to print the shifting of colors due to the hardware. The color images also can apply those methods introduced in the previous parts.

2.3.2 Approach and Procedures

There are two methods can be used: **Separable Error Diffusion** and **MBVQ-based Error Diffusion**.
The **Separable Error Diffusion** method requires to separate an image into CMY (Cyan, Magenta, and Yellow) channels. Then, apply the Floyd_Steinberg error diffusion algorithm to quantize each channel separately.
The **MBVQ** methods requires to determine the MBVQ of the input image first. The following figure shows the algorithm to determine the MBVQ: Second, find the vertex

```
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;
}
```

Figure 20: MBVQ Algorithm

$v \in$ MBVQ which is closest to $RGB(i,j) + e(i,j)$. The following figure 21 shows an example of how to decide color in CMGB Quadruple. Then, compute the quantization

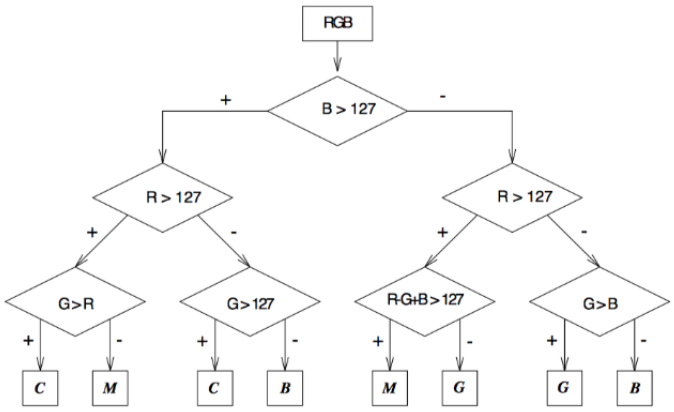


Figure 21: CMGB Quadruple

error $e(i,j) = RGB(i,j) + e(i,j) - v$. Last, distribute the error to neighborhood pixels by Floyd_Steinberg error diffusion.

2.3.3 Result

The following figure 22 is the output image after separable error diffusion with PSNR = 7.915:

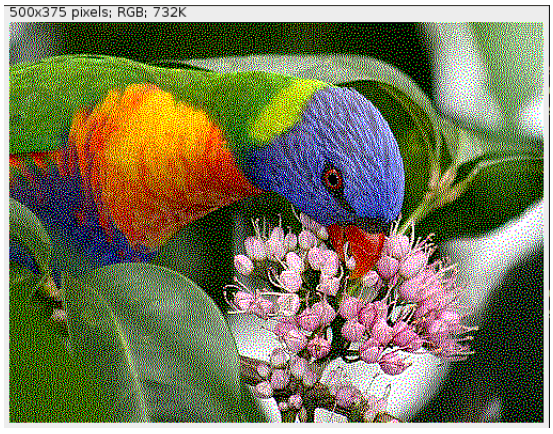


Figure 22: Image After Separable Error Diffusion

The following figure 23 is the output image after MBVQ with PSNR = 7.916:

2.3.4 Discussion

It is clear that there are lost of noise in the half-toned image. One shortcoming of the separable error diffusion method is that the error diffusion filter separately applies to the color channels. Therefore, when combining three channels together, the resulting error will be increased. In addition, the output image is over bright.
Comparing the figure 22 with the figure 23, the MBVQ provides better output images with less noise and more clear color transition. To decrease the error or half-toned noise, the

Figure 23: MBVQ

MBVQ method calculate the error diffusion of three channels in the same time, instead of doing error diffusion separately. The method contains rendering every input color using one of six halftone quadruples and finding the closest color by using quadruples and the original image plus errors. This closest color will be more closed to the original image. As a result, the error during the process will be decreased. In addition, with the help of this quantization process, the brightness of output images will be more nature.

3 Reference

[1] Piotr Dollár and C. Lawrence Zitnick. "Structured Forests for Fast Edge Detection", ICCV (2013).
[2] Structured Edge Detection Toolbox: https://github.com/pdollar/edges
[3] Canny Algorithm: https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection
[4] Doron Shaked, "Color Diffusion: Error-Diffusion for Color Halftones", HP Lab (1999)