USC Viterbi
School of Engineering

DEPARTMENT
Electrical and Computer Engineering

EE 569
Homework #2: Edge Detection and Digital Image Half Toning

Name: Zhiwei Deng
Date: 2/10/2020

Contents

1. Problem 1

1.1 Sobel Edge Detector

1.1.1 Abstract and Motivation

Edge Detection is a technique that used in image processing industry and Machine Learning. The basic method of edge detection is detecting the drastic brightness or color changes between pixels. A pixel possesses a window which contains two or more drastic pixel changes would be considered as an edge pixel. Many methods are developed to complete this task. In this part, a Sobel edge detector will be implemented and tested on two images that shown as below in Figure 1.

(a)

(b)

Figure 1. Test Images (a) Dogs.raw (b) Gallery.raw

In this part, because of Sobel Detector can only detect brightness changes of images, another problem to consider is how to convert an RGB image to a grayscale image which only contains luminance information. I used equation (1) to get the gray images.

$$G = 0.2989 * R + 0.5870 * G + 0.1440 * B \quad (1)$$

Sobel Detector is a type of first order gradient filter, which contains two operators in directions X and Y, which show in Figure 2 as 3 by 3 matrix.

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

Gx

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

Gy

Figure 2. Sobel Operators of Both Directions

1.1.2 Approach and Procedures

For this part, the test image should be converted to grayscale first and then, apply the Sobel operators for each direction to get the gradient values as Gradient-x and Gradient-y. After that, the edge probability file should be generated from the magnitudes of Gradient-x and Gradient-y values using equation (2) as below. The normalization of gradient map is computed by equation (3). At last, a threshold is set to binarize the edge probability and evaluate the performance of the algorithm.

$$Prob = \sqrt{Gx^2 + Gy^2} \tag{2}$$

$$NewProb = \frac{(Prob - Min(Prob))}{(Max(Prob) - Min(Prob))} \tag{3}$$

The workflow graph of the Sobel Detector is shown as below in Figure 3.

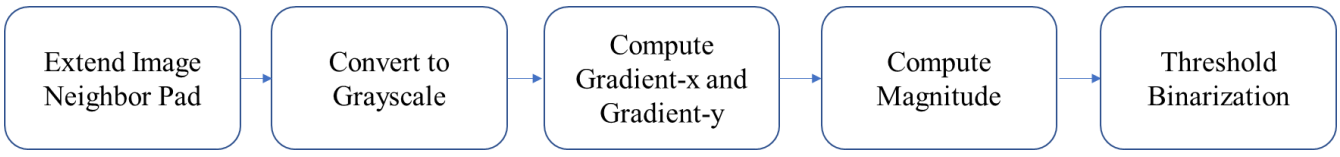


Figure 3. Procedure of Sobel Edge Detectors

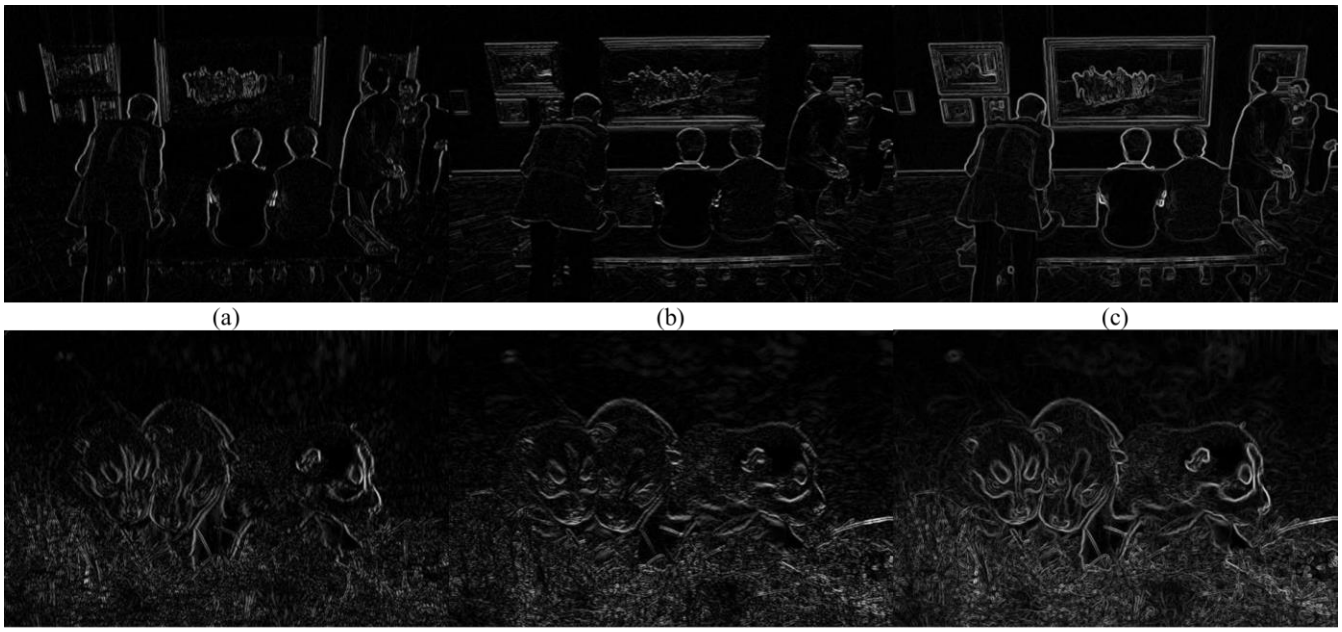
The grayscales of test images are shown in Figure 4.

(a) (b)

Figure 4. Grayscale Images (a) Dogs.raw (b) Gallery.raw

1.1.3 Experiment Result

The gradient maps of Gallery.raw is shown as Figure 5.



(a) (b) (c)

(d) (e) (f)

Figure 5. Gradient Maps X, Y (a-c) Gallery.raw (d-f) Dogs.raw

(a)Gradient-X (b)Gradient-Y (c)Normalized Magnitude Maps

(c)Gradient-X (d)Gradient-Y (e)Normalized Magnitude Maps

The thresholded edge images of Gallery.raw is shown as Figure 6.

The thresholded edge images of Dogs.raw is shown as Figure 7.

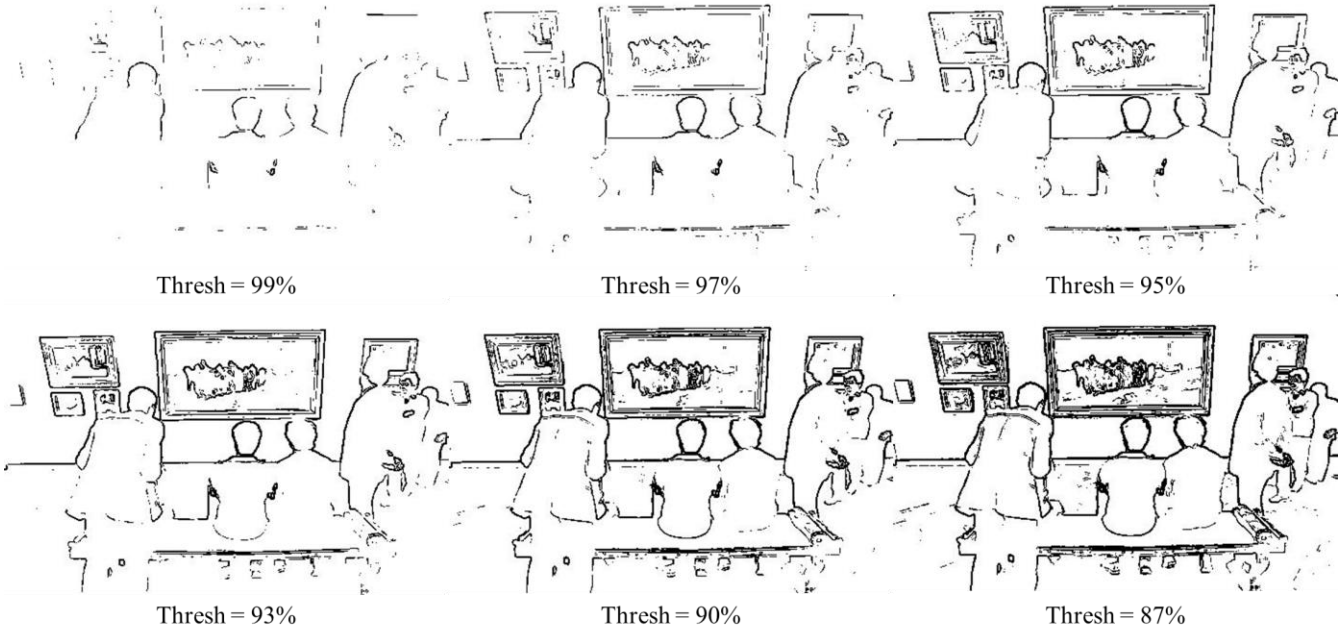


Figure 6. Binarized Edges of Gallery.raw with Different Thresholds

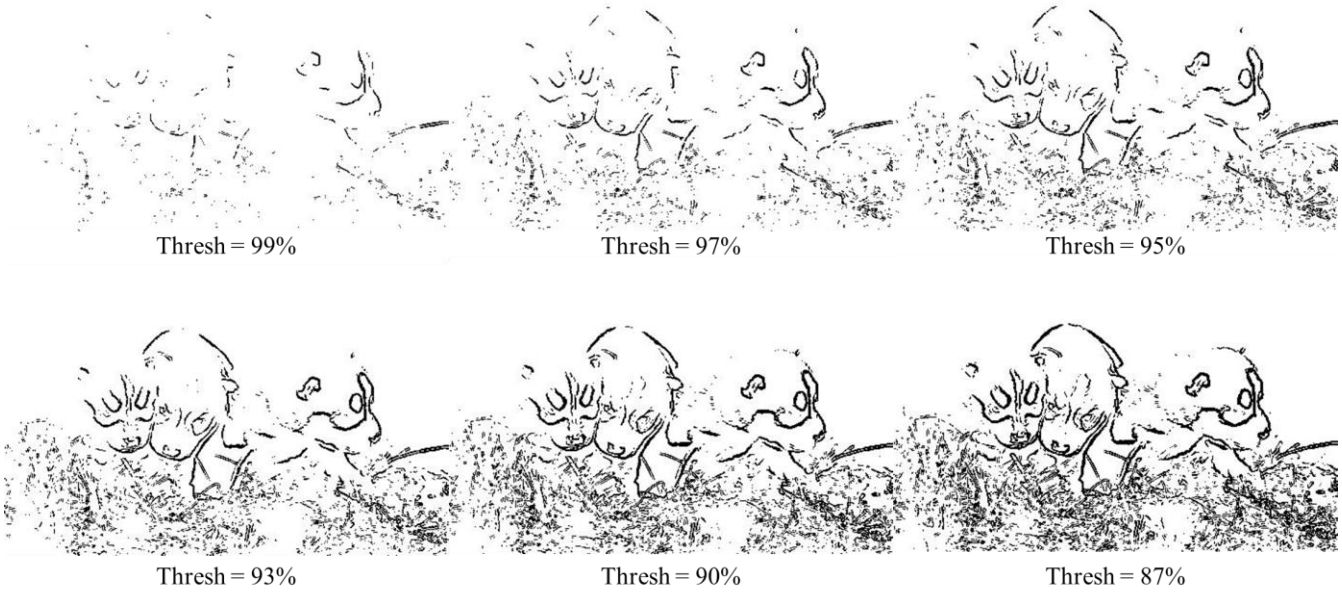


Figure 7. Binarized Edges of Dogs.raw with Different Thresholds

1.1.4 Discussion

Sobel Detector is **one of the simplest first derivative operator** that used in edge detection. But from Figure 6 and Figure 7, we can tell that edges that generated by Sobel Detector is not precise and clear. Many edges are disconnected and cannot filter the backgrounds entirely as shown Figure 8.

Figure 8. Disconnected Edges and Backgrounds

Also, Sobel filter is **very sensitive to noises**, which might cause the un-filtered backgrounds. **To improve its performance, a denoising filter can be applied.** The time complexity of Sobel Operator is *O(m*n*9)(m, n is the width and height of the image)*. **The advantage of Sobel Detector is that it's easy to implement and compute.** For visual quality, we can see that the edge image with threshold = 93% has the best visual quality. For evaluating more precisely, I used F-Score to evaluate each threshold image. The result is shown in Table 1.

Table 1. F-scores of Sobel Detector with Thresholds

Threshold	Gallery	Dogs
99%	0.3036	0.2244
97%	0.6008	0.3804
95%	0.7054	0.4241
93%	0.7459	0.4325
90%	0.7511	0.4257
87%	0.7384	0.4218
85%	0.7241	0.4104

Based on Table 1, we can see that for Gallery image, the best threshold is 90%. And for Gallery image, the best threshold is 93%. The best edge images are shown in Figure 9.

(a) (b)

Figure 9. Best Results of Gallery.raw(a) and Dogs.raw(b)

1.2 Canny Detector

1.2.1 Abstract and Motivation

To improve the performance of edge detection, solving the problem of unfiltered background, disconnected edges and unexpected noises is most concerned. So, Canny detector is proposed. It's also a first order derivative operator and uses double thresholds and non-maximum suppress to generate more clear and connected edges.

In this part, Canny Detector is implemented by Matlab Image Processing Toolbox, and the results' differences between Canny and Sobel will be discussed.

1.2.2 Approach and Procedures

As same as Sobel Detector, we need to pad the image first and convert the image to grayscale, too. Then apply the Canny Detector to find the edge maps of both directions, compute the magnitudes and angles of gradients, and do the NMS and double thresholding at last. The denoising kernel (Gaussian Kernel) is computed by the equation(4).

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{4}$$

The procedure of Canny Detector is shown as Figure 10. The conversion to grascale used the same equation as equation(1).

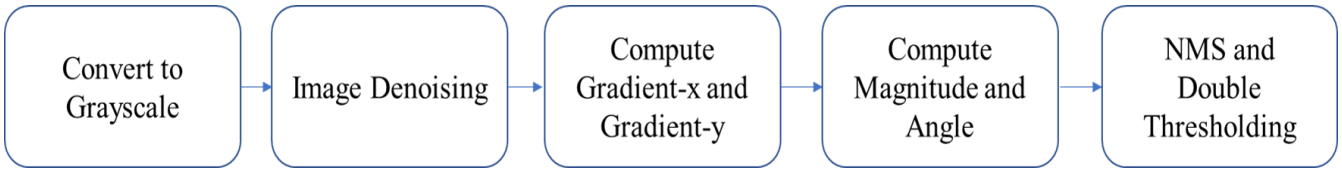


Figure 10. Diagram of Canny Detector Procedure

1.2.3 Experimental Result.

The result images from Canny Detector with different threshoholds are shown in Figure 11 and Figure 12. The threshold pairs are expressed as (a, b), where a is the lower threshold and b is the upper threshold.

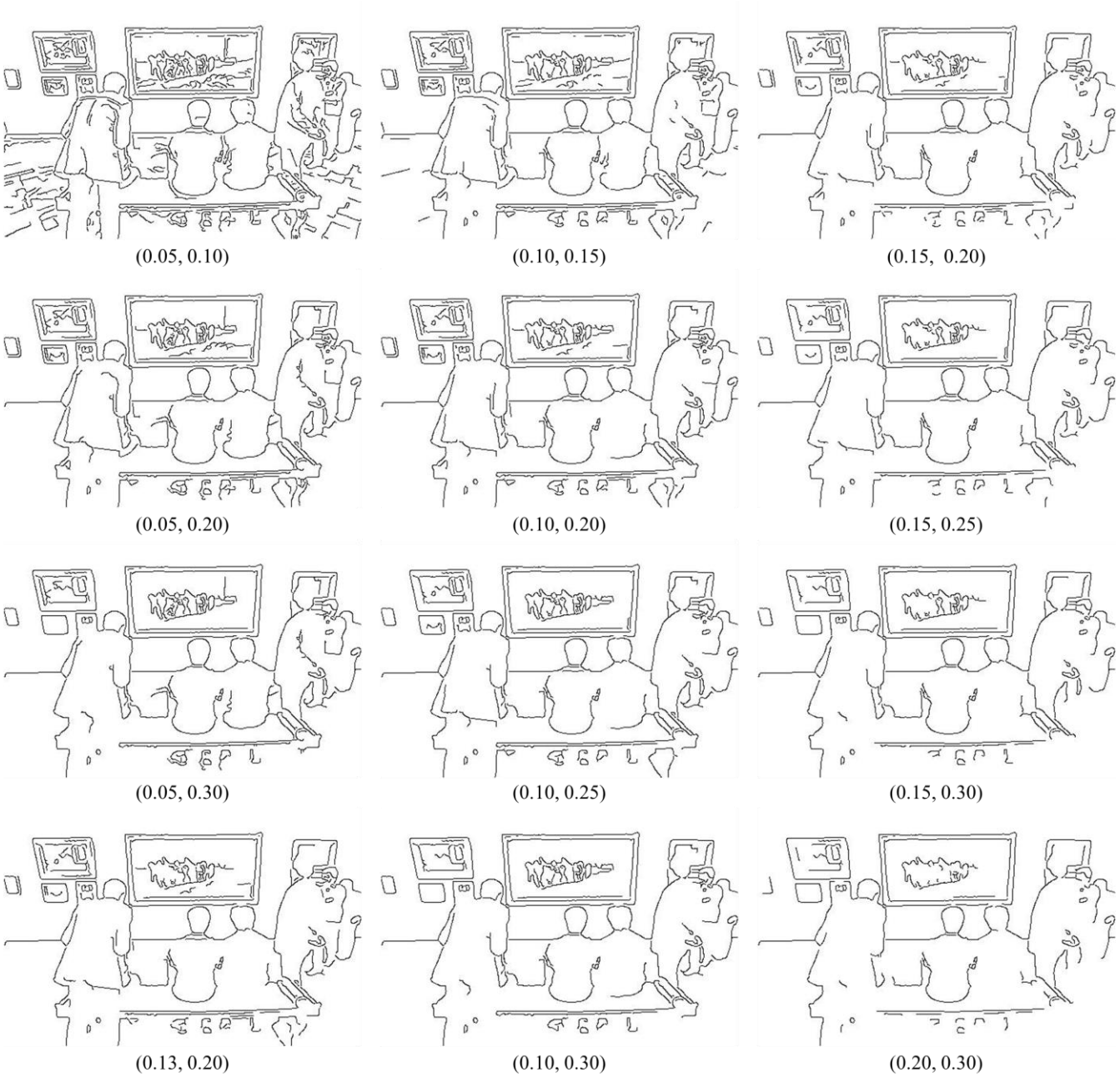


Figure 11. Edge Images of Gallery.raw from Canny Detector

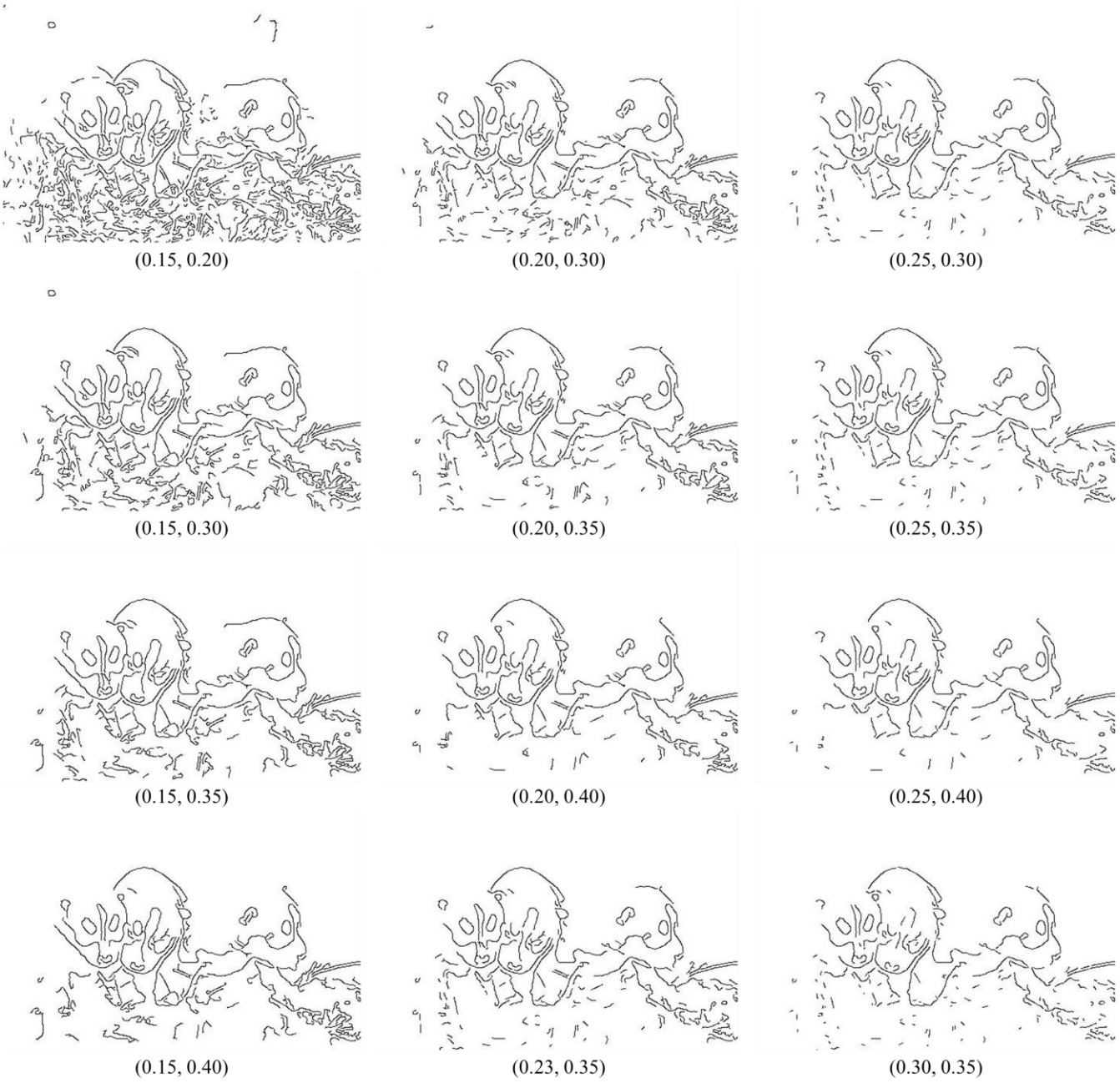


Figure 12. Edge Images of Dogs.raw from Canny Detector

1.2.4 Discussion

The Canny Detector's procedure contains following 4 steps.

Denoising: In Canny edge detection, it uses Gaussian kernel to denoise the image first. The kernel example is shown as Figure 13.

$\frac{1}{2\pi}e^{-1}$	$\frac{1}{2\pi}e^{-0.5}$	$\frac{1}{2\pi}e^{-1}$
$\frac{1}{2\pi}e^{-0.5}$	$\frac{1}{2\pi}$	$\frac{1}{2\pi}e^{-0.5}$
$\frac{1}{2\pi}e^{-1}$	$\frac{1}{2\pi}e^{-0.5}$	$\frac{1}{2\pi}e^{-1}$

Figure 13. Example of Gaussian Kernel N=3,sigma = 1

Differenciating: After the denoising, it uses the first order derivative kernel to compute the gradients of each pixel in both X and Y directions to get the gradient maps as shown in Figure 5. Magnitudes and angles of gradientes of each pixel has also been calculated in this step.

NMS: Because it cannot get edges precisely and clearly only using the magnitude of gradients. It also uses the angle of gradients to check the neighbor pixels to find whether the edge pixel magnitude is local maximum of this window. If the pixel is the local maximum, which means it will be preserved or it will be considered as fake-edge and ignored otherwise.

Figure 14(a) shows the scheme of the NMS, if the red line is the gradient direction, then Canny Detector should check whether pixel E is the local maximum amng pixel a, e and i. For more general conditions when the angle of gradients is not the times of 45 degee as shown in Figure 14(b), we can do the interpolation between a, b and h, I to compare with the e's gradient pixel value.

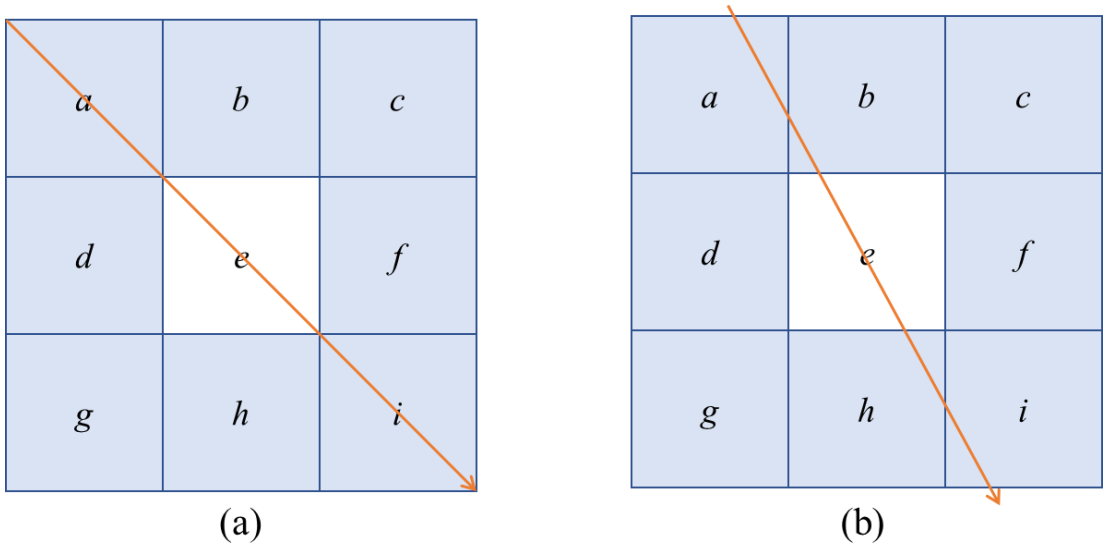


Figure 14. Example of NMS Scheme

Double Thresholding: Canny Detector uses two thresholds to filter the edge probability map. The high threshold is to identify the edge pixels that has large magnitude, and the loe threshold is to filter away the fake edge-pixels which have small gradient magnitudes. For the pixels whose magnitude is between this two thresholds, the detector checks its connectivity to other pixels. This scheme is proposed to assure the connection between close edge pixels and avoid disjointed edges shown in Figure 8.

More precisely, the scheme works as follow.

For M(i, j) > T2, called strong edge pixel, the pixel is kept.

For M(i, j) < T1, called fake edge pixel, the pixel is ignored.

For T1 < M(i, j) < T2, called mid edge pixel. Check if the pixel is next to a strong edge pixel or a mid edge pixel. If so, kept or ignored otherwise.

Effects: We can see from the first column of Figure 11 and Figure 12 that when the upper threshold increases only, the image's details decreases

due to the pixels need to posses larger magnitude to be kept. Otherwise, the edge image will posses too much details which might been considered as fake-edges while the upper threshold is too low. As same as the upper bound, the edges get more clear when the lower threshold increases. Also, when the lower threshold is too big, the edges of results might get disconnected and some strong, correct edge pixels might be filtered mistakenly.

Another thing is that, when the lower bound and upper bound increases at the same time, the result image tends to contain less details and get more simple and strong edges only. While the result tends to have more complex backgrounds and textures which contains more details.

Comparison: In visual quality, Canny Detector's performance is surely much better than Sobel Detector's. The results' edges are thinner and connected and there is no complex grounds left. The problem shows in Figure 8 is well solved in some level. To evaluate the results more precisely, I used the edgesEvalImg function with whole set of Ground Thruths to get Table 2.

Table 2. F-scores of Canny Detector with Thresholds

Threshold	Dogs	Threshold	Gallery
(0.15, 0.20)	0.5177	(0.05, 0.10)	0.7398
(0.15, 0.30)	0.5542	(0.05, 0.20)	0.8051
(0.15, 0.35)	0.5643	(0.05, 0.30)	0.7877
(0.15, 0.40)	0.5508	(0.10, 0.15)	0.8048
(0.20, 0.30)	0.5676	(0.10, 0.20)	0.8179
(0.20, 0.35)	0.5794	(0.10, 0.25)	0.8133
(0.20, 0.40)	0.5774	(0.10, 0.30)	0.7995
(0.23, 0.35)	0.5875	(0.13, 0.20)	0.8188
(0.25, 0.30)	0.5841	(0.15, 0.20)	0.8143
(0.25, 0.35)	0.5839	(0.15, 0.25)	0.8056
(0.25, 0.40)	0.5790	(0.15, 0.30)	0.7851
(0.30, 0.35)	0.5731	(0.20, 0.30)	0.3191

1.3 Structured Edge

1.3.1 Abstract and Motivation

In Canny Detection, if we want to elliminate the backgound noises and only preserves the pure edges, it is a risk to lose the edge pixels as well. While to detect edges more precisely, structured edge detection is introduced. Different from Sobel or Canny, SE Detector is a data-driven approach. It relies on the image datasets and clear ground truth labels to train and perform. Also, the theroy behinds it contains random forest, decision tree and feature extractions.

In this part, the SE Detector will be implemented and the methods and theories will be discussed. Also, the results of SE algorithm will be evaluated and conpared with other detectors.

1.3.2 Approach and Procedures

The structured edge algorithm seen pixel's feature with 32 by 32 patch features, which are going to be labeled based on the relations between each other with a 16 by 16 masks. Then, it uses these features to train and construct the decision tree to the random forest to build the model and use the model to detect the patches are edges or not.

So, the specific procedure of Structured Edge Detector is shown as Figure 15.

Figure 15. Digram of Procedure of SE Detector

1.3.3 Experimental Result

Different from Canny Detector, the out images are not binary but the edge maps with probability within 0~1.

The result images of Gallery.raw from SE Detector with different parameters are shown in Figure 16. The parameter sets are expressed as (multiscale, sharpen, nTreesEval, nThreads, nms).

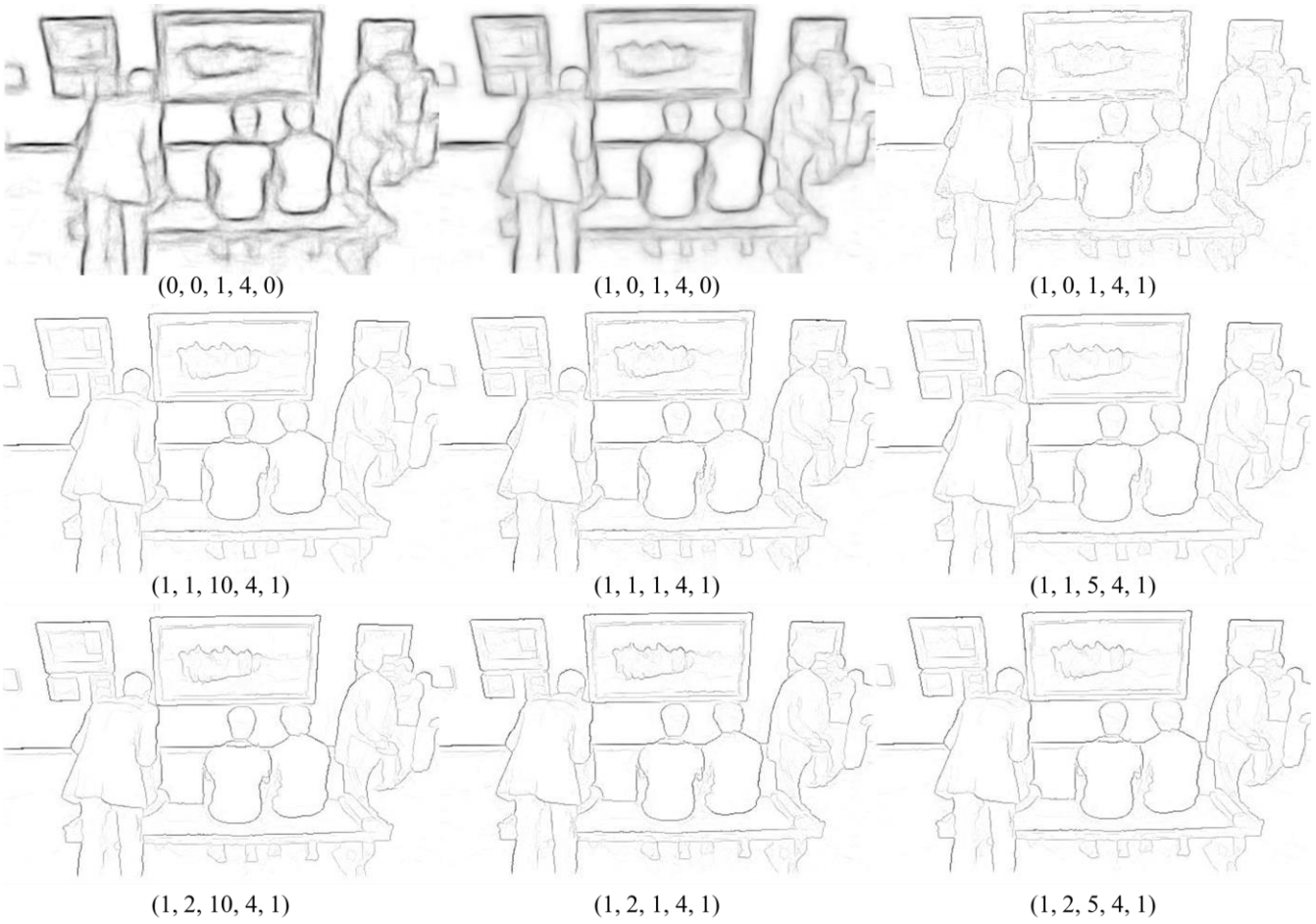


Figure 16. Edge Maps of Gallery.raw from SE Detector

The result images of Dogs.raw from SE Detector with different parameters are shown in Figure 17. The parameter sets are expressed as (multiscale, sharpen, nTreesEval, nThreads, nms).

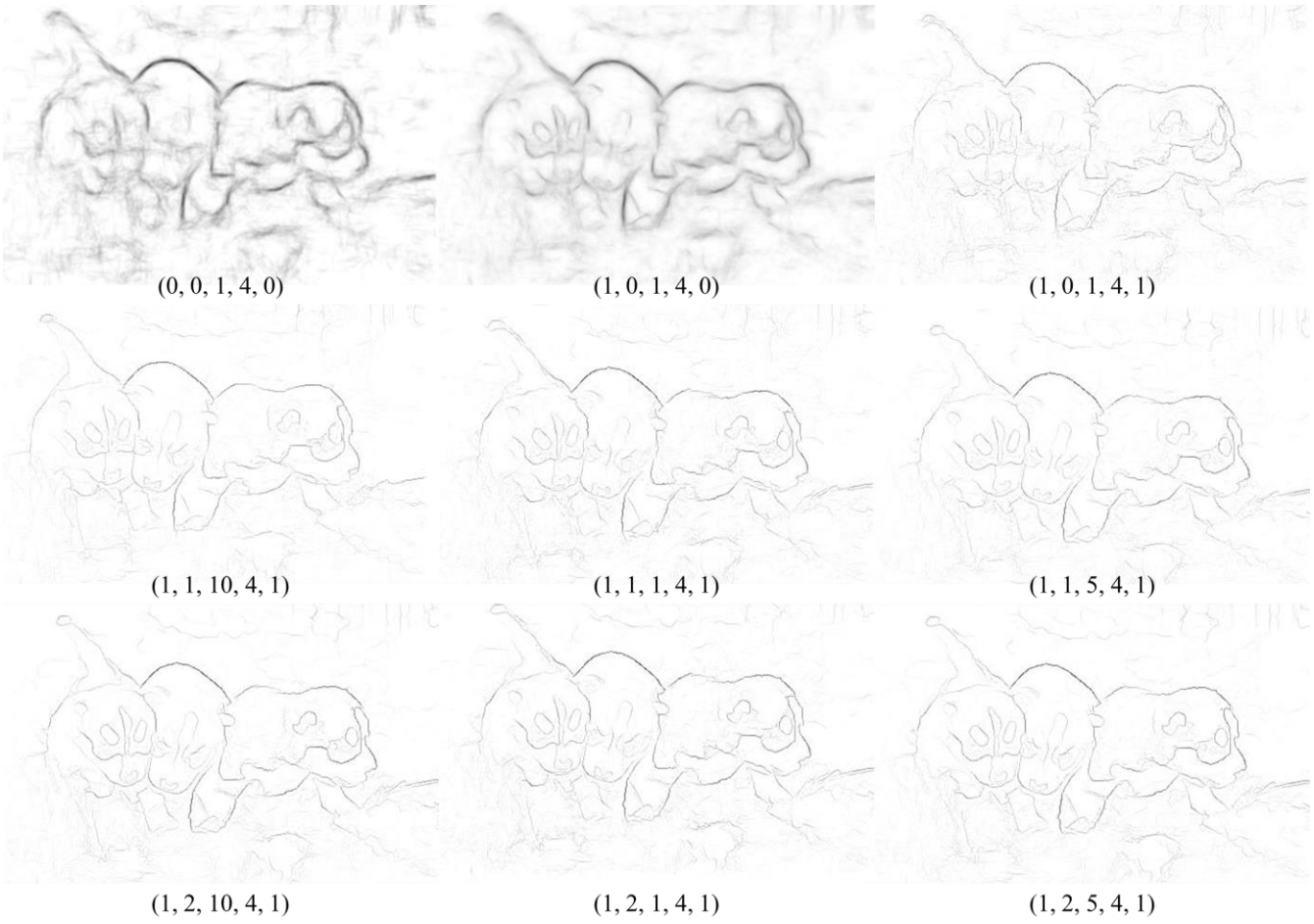


Figure 17. Edge Maps of Dogs.raw from SE Detector

1.3.4 Discussion

Structured Edge Detecor is one of the most advanced and sophisticated operator in image edge detection process. It is trained and tested based on machine learning and image datasets.

Summary: The flow chart of SE algorithm is shown in Figure 18(b). For training the model, SE detector has 4 steps. Firstly, it generates 32 by 32 image patches of the image to be used to train the forest. Then,it will map the image patches to the an intermediate space Z. At last, using the candidate features that generated in space Z. The algorithm randomly trains the decision

trees recursively and constructs the structured Random forest to built the model. The details of each part of the algorithm will be discussed below.

The idea of structured random forest is proposed to optimize the computationally cost due to the large amount of labels and features, we need to cluster the labels and reduce features to trade off computation efficiency.

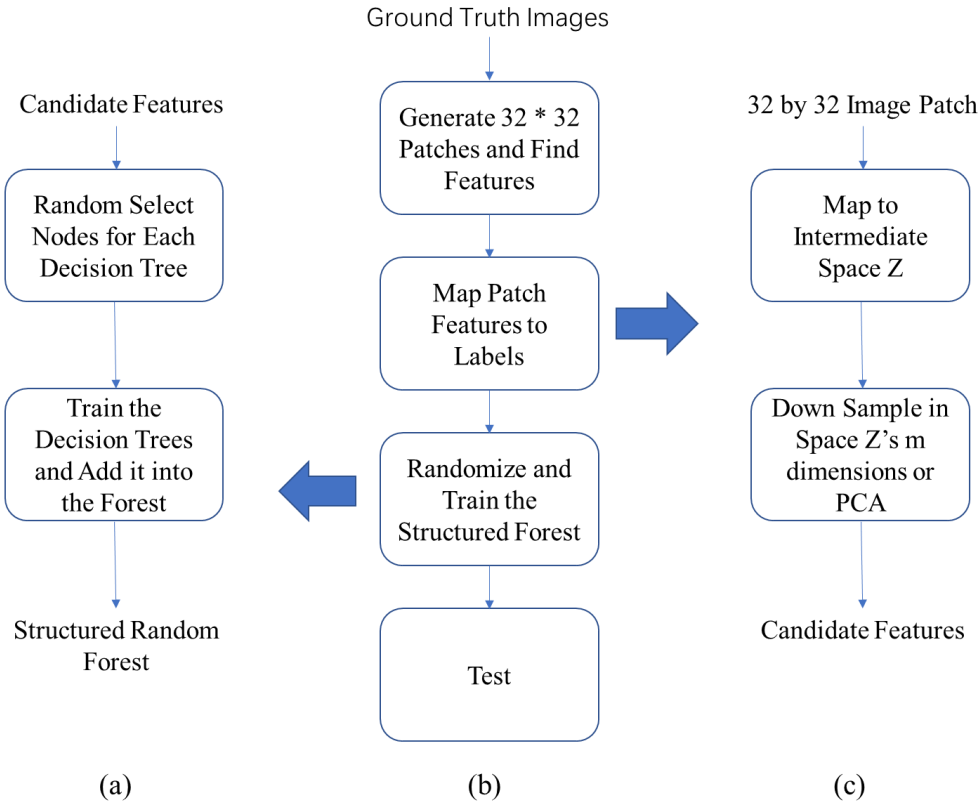


Figure 18. Flow chart of SE algorithm

Features Finding: For every image patch, there are 13 aspects of fetures containing 3 color spaces, 8 directions and 2 magnitude spaces. Firstly, thealgorithm downsampled the image patch with a factor of 2, which would porduce 3328 candidate features for each patch. Then, the algorithm downsampleds these features and computes the differences between them in

each aspect and adds 300 more features in each aspect, which will make the number of features become 7228. To this point, all 7720 candidate features is produced.

Mapping: For the mapping procedure, because the output space of the random forest can be high diemnsional and very complicated. It will be computationally expensive if the forest want to map the candidate features directly to the structured labels. In this case, the output space is 16 by 16 binary masks, which contains 2^{256} possibilities. To calculate and train faster, the intermediate space Z is used. In Z space, the binary values of the vector is only to show whether the segment is same as the other, which can reduce the possibilities to 256C2.

Decision Trees: Decision tree is a machine learning algorithm which can work as a classifier. Decision trees consist 3 types of nodes, root node, parent nodes, and leaf nodes. The leaf nodes indicate the results of the classification. The inside parent nodes and root nodes work as individual classifier. Every input of a decision tree is going to be classified through some parent nodes one by one and settles in one leaf node. The critirer to evaluate a decision tree is called information gain. Which defined by equation(5)

$$I_j = H(\mathcal{S}_j) - \sum_{k\in\{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k) \tag{5}$$

$$H(S) = \sum_y Pylog(Py) \tag{6}$$

In SE Detector, the decision strategy for decision trees is stated below.

Firstly, choose a subset of the patches with selected features, for every decision tree, the algorithm train it from the root node and do the binary classification to find the test and condition which would get biggest information gain. Then, it generates two branches with this two results and recursively apply this two steps in every node of every level in the tree. This process will end when the patches cannot be divided anymore or the size of pathes are too small or the depth of the tree is maximum. And then, a decision tree's construction is completed at this point.

Random Forest: Random forest is a classifier which has multiple decision trees. The problem of the process is that multiple trees will produce multiple outputs, while we only need one. In the principle of random forest, to ensemble the results that predicted by different trees, the RF uses the one with relative majority votes as the final class. Also, there are other ensemble model to aggregate all the predicted results. To introduce the randomness into the forest, RF does not choose the decision tree that produce the best information gain. It chooses randomly from the dataset to form a subset of feature for the sub-tree as stated above and find the optimization of this subsets using certain ensemble model. This randomness can prevent the occurrence of overfit.

Parameter Effects: SE detector has 5 main parameted that we can tune, multiscale, sharpen, nTreesEval, nThreads, nms.

When set mutiscale = 1, the algorithm will apply the detector on all original, half size and double size image. The final results of the detection

will be generated by averaging all three result maps. For visual quality, we can see from the first two images of Figure 16 and Figure 17 that the uninportant edge pixels like painting textures of Gallery.raw and background of Dogs.raw. So, we can say that multiscale has some positive effects to edge detection in some level.

When sharpen is on, it means the algorithm allows the edge pixels to shift from the max magnitude location. And it will adjust the shifted pixels back to true edge location to keep the edges sharp and connected. For example we can compare the third and fifth images of Figure 16, the edges of the painting frame is much more smoother when the sharpen parameter is turned on as shown in Figure 19.

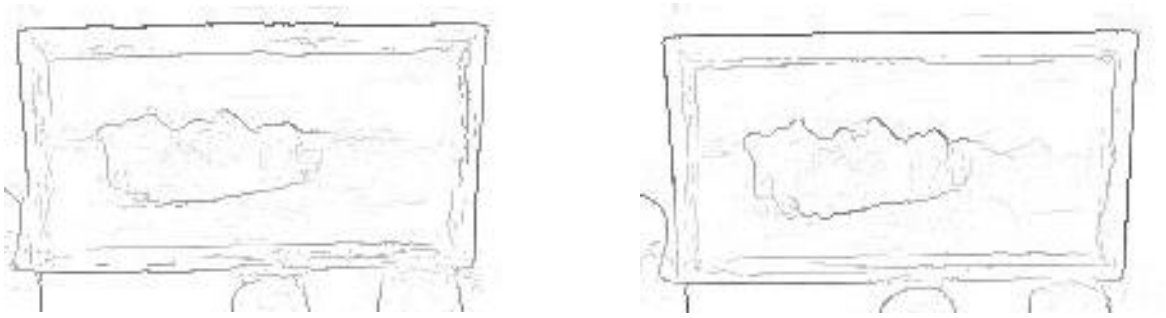


Figure19. Comparison of Between Sharpen = 0 and sharpen = 1

nTreesEval defines the number of decision trees in the Random Forest. Based on Figure 16 and Figure 17, we can tell that along with the inrease of the number, the performance can improve in some level. But there almost no different between the result images of nTreesEval =5 and nTreesEval =10. In another hand, the bigger number will take longer time to compute. So, it is better to set this parameter not too large to trade off the efficiency.

nThreads control the threads usage when running the algorithm. Bigger

nThreads numbrer means more computer resources used, which would reduce the time that the algorithm needs.

Nms controls the non maximum suppression on or not. The method of NMS is disscussed in Canny Detector. We can see from Figure 16 and Figure 17 that non maximum suppression is a very inportant procedure to get thin and clear edegs rather than thick and blurry edges.

Parameter Selection: (1, 2, 10, 4, 1). Based on the discussion above, we can see that NMS, sharpen and multiscale are all inportant to improve the performance of the Detector. For producing more precise image, I choose the biggest random forest with 10 decision trees in it.

Comparison:

The best results that generated by Canny Detector and SE Detecor with Gallery.raw and Dogs.raw are shown in Figure 20.

From Figure 20, we can see that the image generated by SE Detector have more details than Canny Detector, like the body of the right dog and the body of the sitting man are all detected by SE but not Canny. Also, SE Detector is more robust even for an image with complex back grounds, Figure 20(b) has much less fake-edge backgrounds than Figure 20(a). However, the lines of Canny Detector's results are smoother than SE Detector's, which might accounted on the double thresholds.

For the computational complexity, Canny Detector's time complexity is same as Sobel's, O(m*n). While SE Detector's time complexity is same in test process, but the time consumption of training depends on different tasks.

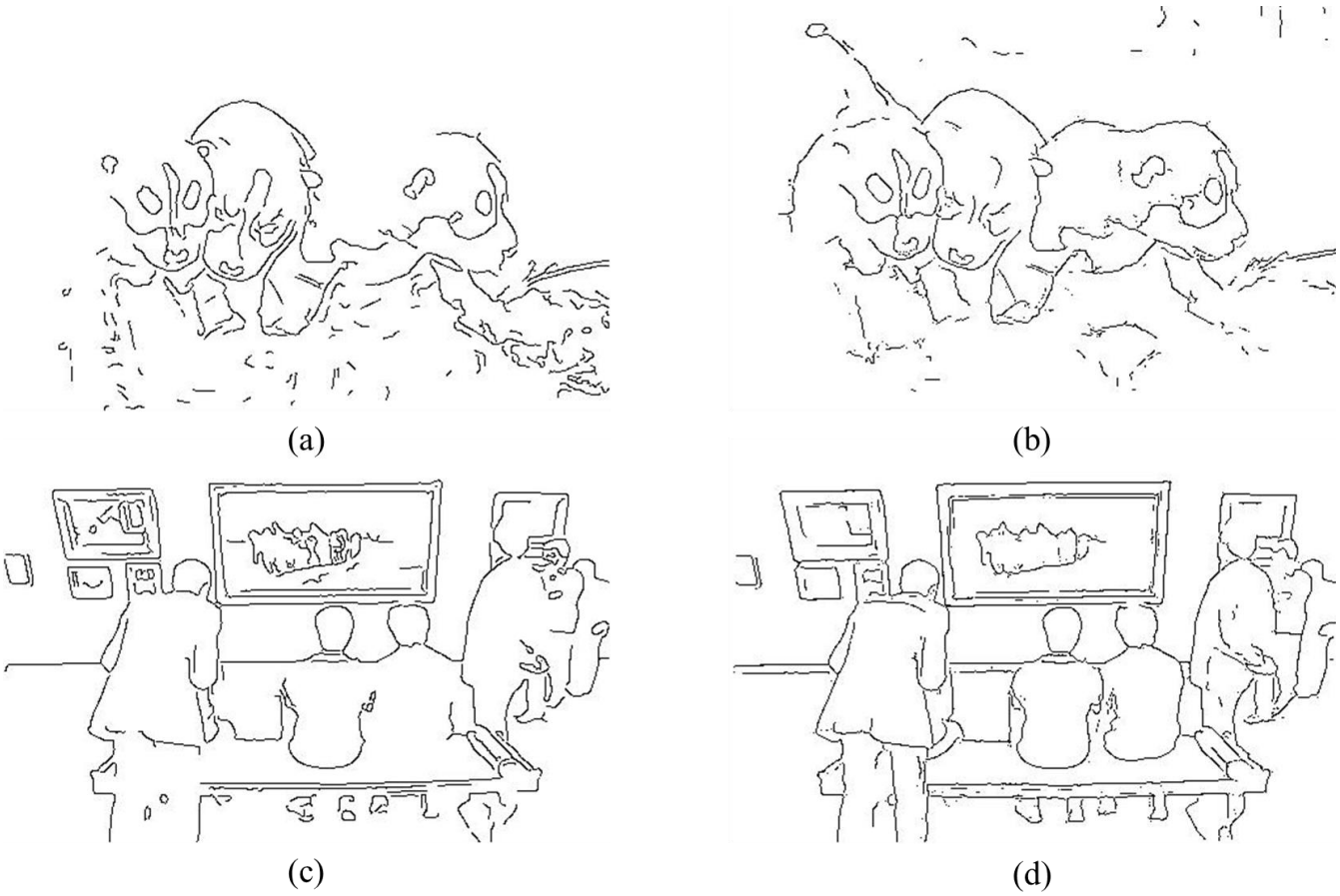


Figure 20. Best Results by Canny(a, c) and SE(b, d) Detector

1.4 Performance Evaluation

1.4.1 Abstract and Motivation

By now, we have implemented three kinds of detectors, Sobel, Canny and SE detector and evaluate them based on visual quality or maximum F-score. The precise evaluation of the results is also important in image processing. F-score is proposed to evaluate binary edge images that generated by different detectors. It is based on the ground truth image.

In this part, the F-measure will be used to evaluate the performances of all detectors by using the Structured Edge toolbox in Matlab. The F-measure and the performances of detectors will be compared and discussed.

1.4.2 Approach and Procedures

For the evaluation of detectors, we use F-measure to define the performance of each results. Higher F-score means higher quality of edge detection. F measure is computed by precision and recall, which defined as follow in equation(7-9).

$$P = \frac{\#True\ Positive}{\#True\ Positive+\#False\ Positive} \tag{7}$$

$$R = \frac{\#True\ Positive}{\#True\ Positive+\#False\ Negative} \tag{8}$$

$$F = \frac{2*P*R}{P+R} \tag{9}$$

Using F-meaure can prevent the result image to have a biased judgement such as too few edges without backgrounds or too much details with complex background. The procedure of the algorithm is described as pseudo code as shown in Figure 21.

```
for each GT g:
    for each threshold t:
        compute Precision(g,t) and Recall(g,t)

for each GT g:
    compute mean_Precision_over_thresholds(g), mean_Recall_over_thresholds(g) #"mean precision, recall for each ground truth"
compute mean_F = 2 * mean(mean_Precision_over_thresholds) * mean(mean_Recall_over_thresholds) /
                    (mean(mean_Precision_over_thresholds) + mean(mean_Recall_over_thresholds))

for each thredhold t:
    compute mean_Precision_over_GTs(t), mean_Recall_over_GTs(t)
    compute F(t) = 2 * mean_Precision_over_GTs(t) * mean_Recall_over_GTs(t) / (mean_Precision_over_GTs(t) + mean_Recall_over_GTs(t))
plot F #x-axis would be threshold values
find the best F score max(F)
```

Figure 21. The Pseudo Code for Evaluation

Different from the Table 1 and Table 2 shows, in this part, I used the average perforance for each ground truth instead of maximun performance overall.

1.4.3 Experimental Results

For clearer to compare with each other, I show the results in tables for each detector. The performance of best Sobel Detector image is shown as Table 3.

Clarify: In this step, because the evaluation method is different from the ones used in Table 1 and Table 2. Previously, I used edgeEvalImg function to evaluate the Full GT cells at the same time. The codes in the function might be average the Precision and Recall in a different way rather than uniform average. But in this part, I used the uniform average among thresholds for each GT and among each GT to get a final score.

Also, because the inside evaluation procedure might have changed. The best results image may change as well. And for Canny detector, because the thresholds are a part of the detector, so the thresholds don’t change and as shown in Figure 20 (a, c). For those detectors whose best image has changed, I will mark the parameters again.

Table 3. Sobel's Performance Evaluation

Overall Best T = 0.13(87%) for Gallery.raw, Overall Best T = 0.31(69%)for Dogs.raw

Sobel	Ground Truth	Precision	Recall	F-Score
Gallery	GT1	0.30468	0.66107	0.41712
	GT2	0.32098	0.70662	0.44144
	GT3	0.30195	0.62684	0.40757
	GT4	0.32857	0.65857	0.43841
	GT5	0.30657	0.63923	0.41440
	Overall	0.54530	0.69400	0.61070
Dogs	GT1	0.30349	0.15624	0.20628
	GT2	0.21636	0.177221	0.19484
	GT3	0.29542	0.18013	0.22380
	GT4	0.29984	0.14368	0.19427
	GT5	0.32409	0.31285	0.31837
	Overall	0.19920	0.44220	0.27470

Table 4. Canny's Performance Evaluation

Overall Best T1 = 0.13, T2 = 0.20 for Gallery.raw, Overall Best T1 = 0.23, T2 = 0.35 for Dogs.raw

Canny	Ground Truth	Precision	Recall	F-Score
Gallery	GT1	0.81294	0.54799	0.65467
	GT2	0.83719	0.50363	0.62892
	GT3	0.80997	0.51877	0.63246
	GT4	0.84437	0.50182	0.62951
	GT5	0.83353	0.73751	0.78259
	Overall	0.56190	0.82760	0.66940
Dogs	GT1	0.64664	0.25438	0.36513
	GT2	0.34411	0.28132	0.30956
	GT3	0.49933	0.28590	0.36361
	GT4	0.61864	0.20407	0.30690
	GT5	0.74446	0.55527	0.63610
	Overall	0.3162	0.5706	0.4069

Table 5. SE's Performance Evaluation

Overall Best T = 0.18 for Gallery.raw, Overall Best T = 0.14 for Dogs.raw

SE	Ground Truth	Precision	Recall	F-Score
Gallery	GT1	0.39314	0.71846	0.50820
	GT2	0.41376	0.70258	0.52081
	GT3	0.38679	0.62178	0.47691
	GT4	0.40416	0.61078	0.48644
	GT5	0.36230	0.79169	0.49711
	Overall	0.6781	0.87680	0.7647
Dogs	GT1	0.26977	0.49125	0.34828
	GT2	0.16421	0.51246	0.24872
	GT3	0.20511	0.49967	0.29084
	GT4	0.27688	0.46240	0.34636
	GT5	0.23849	0.62530	0.34528
	Overall	0.49140	0.6746	0.56860

1.4.4 Discussion

Comparison between Detectors: From Table 3 to Table 5, we can see clearly from the overall F -measure that the performance of SE > Canny > Sobel.

For Sobel detector, because there is no NMS in it, it is sensitive to noises and complex backgrounds. That is why the Dogs.raw image has a much worse performance than Gallery.raw. Also, the single threshold is rather too simple to check and filter the edge pixels. But for the pros of Sobel operator is that it easy to implement and fast to compute, which makes it suitable for some real time processing without high requirements of quality.

For Canny detector, because there is NMS scheme in it, it can filter some complex backgrounds and noises. That is why the Dogs.raw image

improve so much and Gallery.raw only increases a little. The NMS keeps the detector from the noises, although it is easily to misjudge some complex background pixels as edge pixels. Some denoising method could be applied in the preprocess to weaken the background details and keep main edges.

For SE detector, the performance has a significantly optimized compared with Sobel and Canny, in both F-scores and visual quality. The SE detector is the most robust one among all three of them, its performance with simple or complicated background are all reliable. But there is another thing to address is that SE needs more resources to train the data and more complexity to test the image.

Comparison between Images: From the three tables above, we can see that the F-measure of Dogs.raw is generally less than Gallery.raw under the same detector. This is due to its more complicated textures and backgrounds. For example, the paintings and the body of the persons in Gallery.raw are relatively simple and pure, there is no face in the image, too. But the grass field, the fur of the dog and the sunshine has much more details than elements in Gallery.raw, which might cause more fake-edge pixels to be labeled. This will increase the number of false positive pixels, which will cause a lower precision and a lower F-score.

Precision and Recall: For each image, we can get the graph of the F-score and the thresholds for optimize the results and find the best threshold. The graph can be plotted as below in Figure 22. I used Sobel and Gallery.raw as an example, the graph of SE is quite similar.

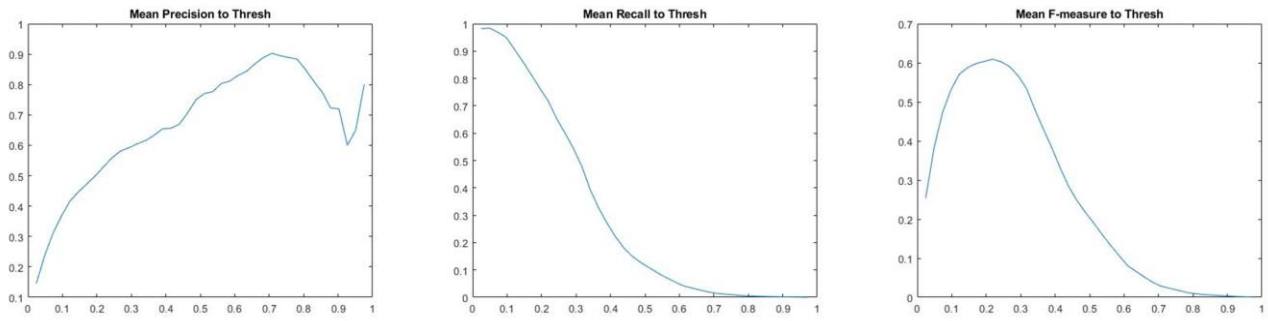


Figure 22. Precision, Recall and F-measure with Thresholds

Rationale of the F-measure: Precision and recall cannot identify the performance of edge Detector separately. For example, if we want to get a high precision, we should just label fewer and fewer pixels as true positive, the precision will peak due to there is less and less false positive pixels. The same reason for recall. From the equation (9), we can see that F value is affected more by the smaller value between Precision and Recall. As shown in Figure 23. The slope near to zero is much bigger than the slope near to 1. For certain P + R, we can see from the figure that it is only when the P = R, the F score can reach the highest.

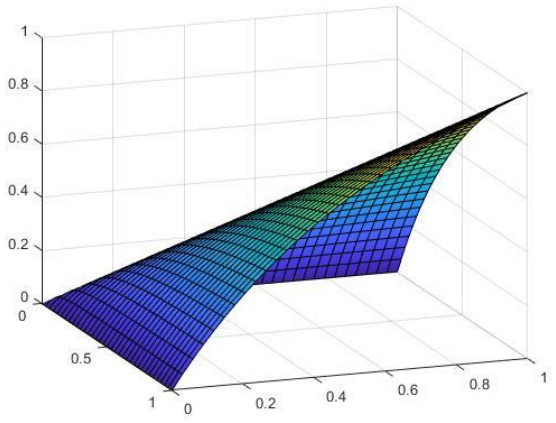


Figure 23. Relations Between F and P, R

2. Problem 2

2.1 Dithering

2.1.1 Abstract and Motivation

Halftone is a technique that widely used in printing industry. It represents the continuous image by certain amount of pixel values and colors. Cause for printer, it will be difficult and inefficient to have all kinds of inks corresponding each pixel values. The simulation of images using limited values is necessary and practical.

Dithering is one of the practical method that converts graysclae pixel values into binary value. There are different types of dithering. I will implement 3 methods in this part and their performances will be discussed and compared.

2.1.2 Approach and Procedures

Fixed Thresholding: The grayscale image pxiel will be set to 0 if the pixel intensity is less than certain threshold T or will be set to 1 otherwise. The procedure of this method can be explained by equation(10).

$$G(i,j) = \begin{cases} 0 & if\ 0 \le F(i,j) < T \\ 255 & if\ T \le F(i,j) < 256 \end{cases} \quad (10)$$

Random Thresholding: In order to break the monotones of the fixed threshold. Random thresholding is introduced. For each pixel of the image, generate a random number between 0 to 255, rand(i, j). The grayscale image pxiel will be set to 0 if the intensity is less than rand(i, j) or will be set to 1 otherwise. The procedure of this method can be explained by equation(11).

$$G(i,j) = \begin{cases} 0 & if\ 0 \le F(i,j) < rand(i,j) \\ 255 & if\ rand(i,j) \le F(i,j) < 256 \end{cases} \quad (11)$$

Dithering Matrix: Instead of generate random methods and introduce noises into the original image. We can generate a matrix to compare the pixels with the matrix elements. The matrix indexes indicates the probability of the pixel to be turned on. The number is larger, the less probability it will be turn on. Also, the dithering matrix is generated recursively, which means I4 is generated based on I2 and I8 is generated based on I4. The conputation of the dithering matrix and the binarization of pixels is shown as below in equation (12-15).

$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} \quad (12)$$

$$I_{2n}(i,j) = \begin{bmatrix} 4 \times I_n(i,j) + 1 & 4 \times I_n(i,j) + 2 \\ 4 \times I_n(i,j) + 3 & 4 \times I_n(i,j) \end{bmatrix} \quad (13)$$

$$\mathrm{T(x,y)} = \frac{I(x,y)+0.5}{N^2} \times 255 \quad (14)$$

N denotes the width of the matrix. Because the image is reasonably bigger than the matrix, the indexes of the matrix are going to be used periodically.

$$G(i,j) = \begin{cases} 0 & if\ F(i,j) \le T(i\ mod N, j\ mod\ N) \\ 255 & else \end{cases} \quad (15)$$

There is no diagram for these three methods cause the formulas are quiet straight forward and simple.

2.1.3 Experimental Result

The result images after these three methods are shown in Figure 24.

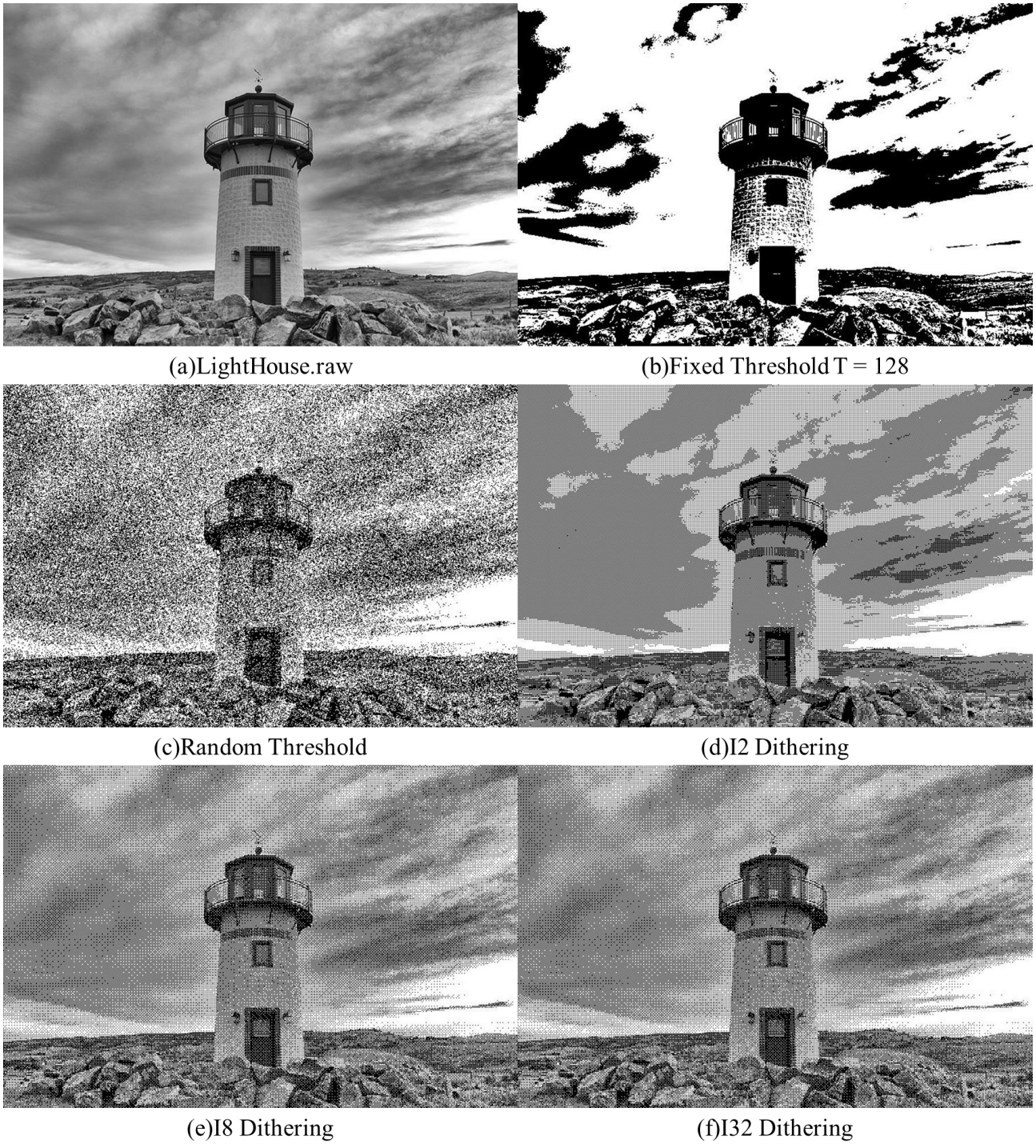


Figure 24. Results After Fixed and Random Thresholding and Dithering

2.1.4 Discussion

For fixed thresholding, the result image only has two kinds of pixels, 0 or 255. Many regions appear to be pure white or black. Details of the image are mostly gone, the textures of the clouds and the door can hardly be recognized. The method is very easy to implement and fast to run, but the result is not acceptable for most printing conditions. Of course, this binary image is useful in some certain contexts, but this method is not a satisfying way to print anything in general.

For random thresholding, there are much more details are kept than fixed thresholding. There is no pure white or black regions in the image,which preserves more details and more accpetable to hunman eyes. While the speed of this algorithm is almost as fast as fixed threshold, the result image is acceptable in some contexts, where the details and the printing resolution is not strictly required. But this image would change the original information of the image by introducing the random noise, which we can tell from the result image. Since the grayscale values are binary, we cannot apply a denoising filter on the result without change the pixel values. In other words, the random noises cannot removed by denoising procedure.

For dithering matrix, the images generated by DM are more clear and similar to the original image compared to the fixed thresholding and random thresholding. Also, unlike random thresholding, DM algorithm is more stable, all regions are as clear as each other. The details and textures of the image is preserved well. From Figure 24, we can tell that the performance of DM

increases while the width of the matrix increases. Specificly, the performances ranked by I32 > I8 > I2. The I8 image and the I32 image is quiet similar, however the construction of the matrix with width of 32 is more complicated than I8. So, in order to trade off efficiency, a medium size window of dithering matrix might be better to implement in practice. The disadvantage of DM is that it posses some square periodical pattern texture due to the periodically crossing the matrix.

To compare these methods more clearly, the pros and cons of these three methods are listed in Table 6.

Table 6. Comparison of Fixed, Random Thresholding and Dithering Matrix

Fixed Thresholding	Pros	Fast to compute and easy to implement.
	Cons	Not acceptable result. Few details.
Random Thresholding	Pros	Relatively acceptable result. Fast to compute and easy to implement.
	Cons	Introduces random noises. Noises cannot be removed by denoising.
Dithering Matrix	Pros	Details preserves more. Easy to implement. Stable and can manipulate the performances.
	Cons	Introduces unexpected periodic pattern.

2.2 Error Diffusion

2.2.1 Abstract and Motivation

Apart from the methods we have discussed. There is another way to halftone a grayscale image. This method is called error diffusion, whose main idea is distributed the errors of the current pixel into the neibour pixels. By doing this, some pixels around the threshold T might be what it deserves to be. For example, with a threshold of 128, a pixel of 125 that surrounded by pixels with value of 200 is going to be 255, too. The disjoint region and impulse noise are filterd by error diffusion.

In this part, a error diffusion method is implemented and three filters are used. The performances of the algorithm and the filters will be compared and discussed.

2.2.2 Approach and Procedures

Error difusion's procedure is as well straight forward. The algorithm will loop over the whole image with a serpentine scnning. For each pixel of the image, check if it should be set to 255 or 0. Then compute the difference between the final value and the original vlaue. Apply the filter to the pixel, which will distributed its errors with different weights. Do this three steps until the loop is over. For each filter, we only apply it to the future pixels, which means we should apply the original filter in odd row, and apply the reflected filter in even row(in C++ syntax, exchange the order).

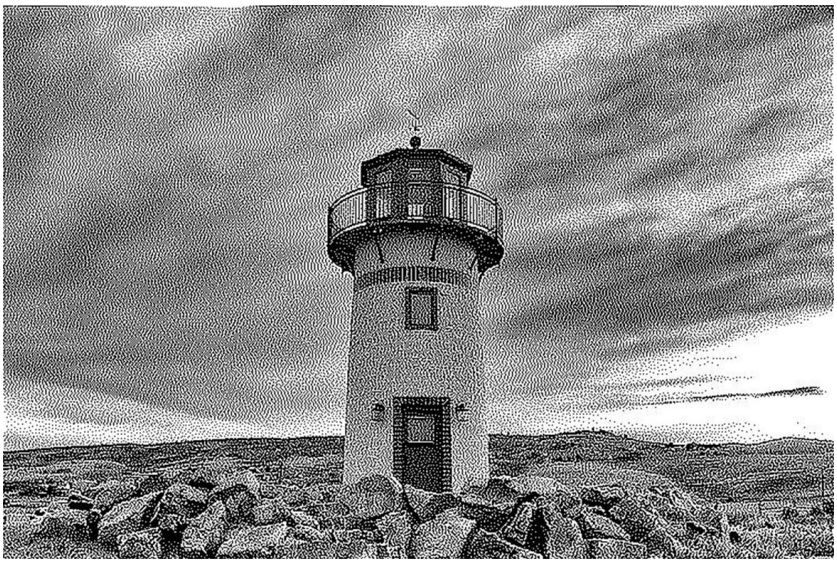
The Flyod-Steinberg's matrix, the JJN matrix and the Stucki matrix is shown in Figure 25.

$$FS = \frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix} \quad \text{JJN} = \frac{1}{48}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix} \quad \text{Stucki} = \frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

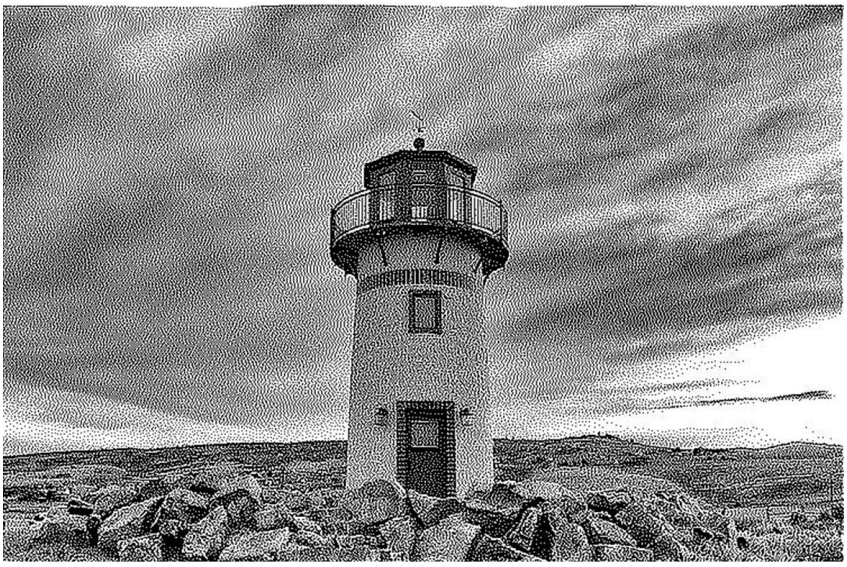
Figure 25. Three Diffrerent Filters

2.2.3 Experimental Result

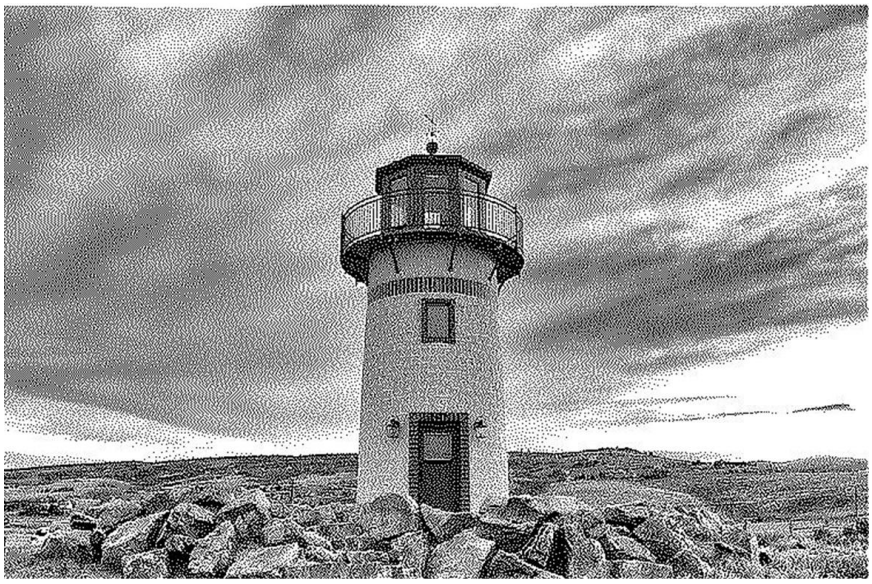
The result images after error diffusion halftoning is shown in Figure 26.



(a)Floyd-Steinberg



(b) JJN



(c)Stucki

Figure 26. The Results After Error Diffusion Halftoning with T = 127

2.2.4 Discussion

The three results of different filters are almost same. In practice, I would like to use Floyd-Steinberg Matrix. Because it is more efficient and faster than the other two due to its smaller kernel.

Comparison: From the Figure 24 and Figure 26, we can say that the performances of error diffusion matrixes are much better than Fixed thresholding, Random thresholding or Dithering Matrix. More details are preserved, such as the tip of the light house and the edges of the rocks. As stated above, the error difusion process prevents the wrong pixel judgments around the threshold, which might result in this performance improvement. However, the result images of error diffusion contain some unexpected pattern like DM do. Instead of periodic square pattern, some parts of the image have patterns and textures like water and maze, as shown in Figure 27.

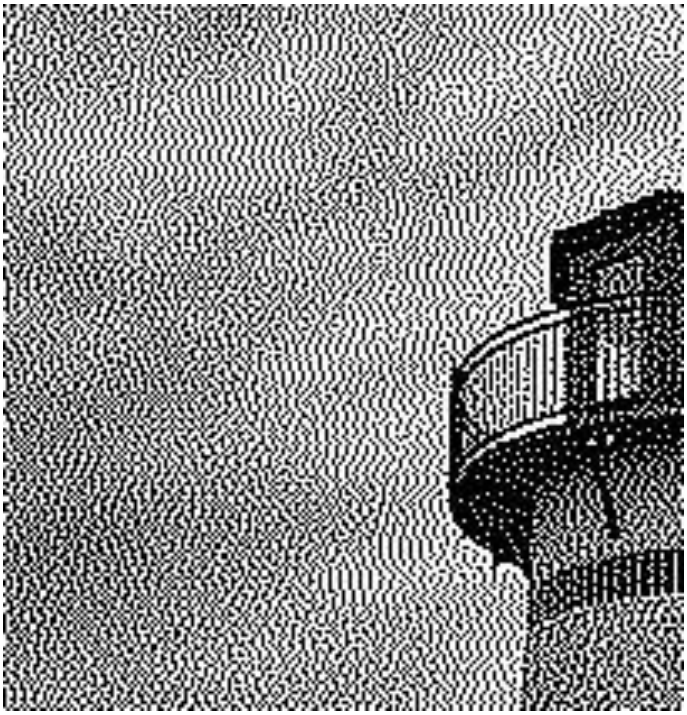


Figure 27. Maze Patterns in Floy-Steinberg Result Image

Improvement: To prevent the maze pattern and the water pattern of current error diffusion algorithm, we can change the fixed thrshold to an adaptive one. We can calculate the mean intensity of the neighbours of the current pixel of current window as well as last window to decide the current threshold for current pixel. This will prevent on pixel is too small or large to affect the neighbour pixels, which might be helpful to reduce the unexpected patterns. Anther way is to improve the performance is that we can only distributed 75% of errors instead of all errors to the neighbour pixels. This approach would reduce the contrast in the pure black or white regoin but enhance the contrast of the mid-intensity region, which might be helpful to get a more clear and enhanced image.

2.3 Color Halftoning with Error Diffusion

2.3.1 Abstract and Motivation

We have discussed that error diffusion has a much better performance than dithering methods with grayscale intensity images. The next reasonably thought is how about the application of error diffusion with color images. To solve the color printing problem, there are two different ways to do this task. Seperately apply the grayscle error diffusion to the R, G, B channels of the image or use MBVQ methds.

In this part, I will implement these two algorithms and apply it to the Rose.raw image. Their performances will be discussed and compared. The Rose.raw image is shown in Figure 28.

Figure 28. Rose.raw

Note: The MBVQ algorithm is implemented by myself using C++.

2.3.2 Approach and Procedures

The seperable application of error diffusion is straight forward cause it is just the extension of grayscale error diffusion three times. Instead of computing the error and do the diffusion in the RGB space, it is done in thr CMY space. The transform between RGB space and CMY space is shown as equation(12) and equation(13).

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \tag{13}$$

For MBVQ method, firstly we need to locate the pixel into one of the six pyramids based on the R, G, B value of the pixel. Then using six decision trees to find the corresponding vertice of current pixel. There are eight useable pixel values, white, yellow, cyan, magenta, green, red, blue and black. These colors is the vertices of the RGB cube. Then it distributed the error of each channel into the neighbour pixels using Floyd-Steinberg, JJN or Stucki filter. In summary, the procedure of MBVQ is shown in Figure 29.

Figure 29. The Diagram of MBVQ Procedure

2.3.3 Experimental Result

The result images after separable error diffusion and MBVQ error diffusion is shown in Figure 30.

(a)Floyd-Steinberg Separable

(d)Floyd-Steinberg MBVQ

(b)JJN Separable

(e)JJN MBVQ

(c)Stucki Separable

(f)Stucki MBVQ

Figure 30. The Results After Separable and MBVQ Error Diffusion T = 128

2.3.4 Discussion

Shortcoming of Seperable Error Diffusion is not obvious if not zoomed in. If the seperable results are zoomed in, we can see many white dots in the image, which means a sudden change in brightness. This shortcoming of the seperable method is caused by the ignorance of the correlations of three channels and the luminance variance of the image. Since the visual system of human is sensitive to the intensities, the sundden change of pixel values might considered as noises. The differences between separable method and MBVQ is shown in Figure 31. We can see that the MBVQ method(right) has much fewer white dots in the red circles, which will produce a smoother visual quality.

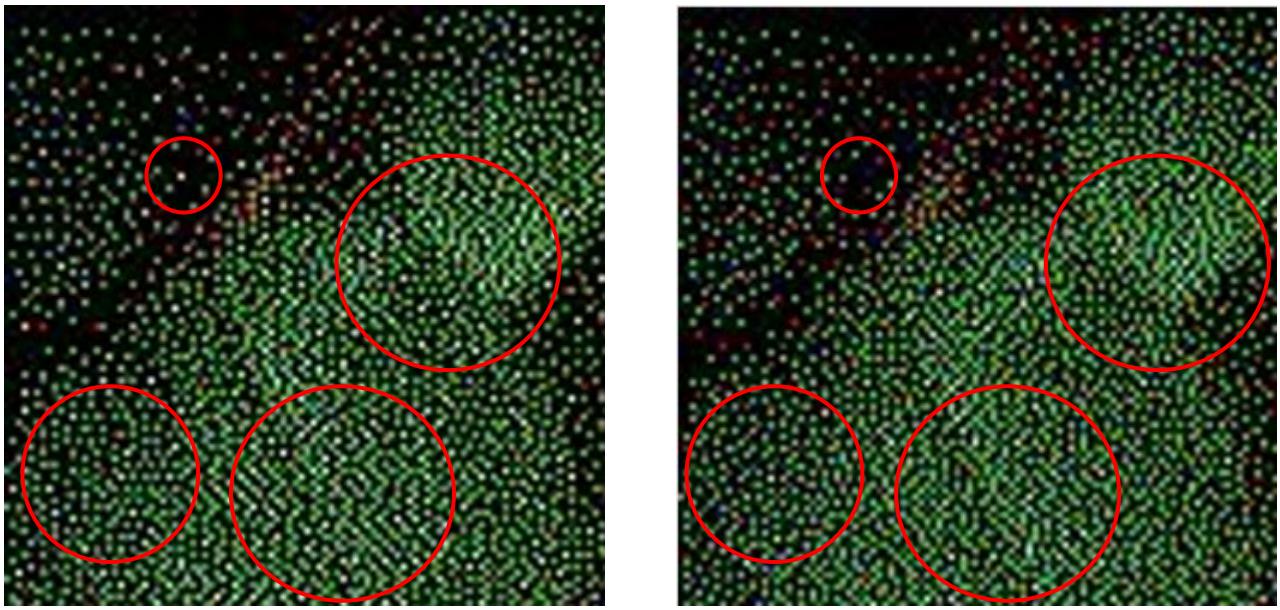


Figure 31. Comparison bewttwen Separable and MBVQ method

Key Idea of MBVQ: The MBVQ can overcome the shortcoming of the separable method because it counts the correlation of the channels into the results. Firstly, it locates every pixel into one of 6 pyramids which represents

the strength of brightness. And using a more complicated scheme, decision trees to find the best-fit color from the palette, which will cause the mininmum brightness change. So, for one region of the image, the differences of brightness of pixels are reduced and the pixels' value cannot change suddenly easily, which reduce the white and black dots of the image. We can say that the MBVQ uses pyramid and decision trees to find the vertices for pixels which will produce minimum luminance variance.

Comparison: Besides the image showns in Figure 28, in general we can see that the result image of MBVQ is slightly more colorful than the result produced by separable method, especially in the center of the roses. Of course the edges and the petals of MBVQ are more smooth and clear due to the preservation of of the brightness. For more straight forward representation, the regional comparison is shown in Figure 32. We can see less ditortion of the flower petal and more vivid rain drops from MBVQ results(right).

Figure 32. Comparison bewttwen Separable and MBVQ method

Of course, because the seperable method and MBVQ both are based on the error diffusion matrix and algorithm, from Figure 31 and Figure 32, we can see some maze like patterns in both result images.

For complexity, the time complexity of the MBVQ and Separable method both is O(m*n). Although the MBVQ is a little harder to implement, its performance improvements deserve the more work. So, in practice MBVQ should be more preferrable.

References

[1] J. Canny, “A computational approach to edge detection,” IEEE Transactions on pattern analysis and machine intelligence, no. 6, pp. 679–698, 1986.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 898–916, May 2011.

[3] P. Dollár and C. L. Zitnick, “Structured forests for fast edge detection,” in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1841–1848.

[4] D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones", HP Labs Technical Report, HPL-96-128R1, 1996.