EE569 Introduction to Digital Image Processing

Homework Report #2

Name: Boyang Xiao **USC ID:** 3326730274 **Email:** boyangxi@usc.edu

I. Problem 1: Edge Detection

1.1. Part a: Sobel Edge Detection

1.1.1. Abstract and Motivation

Edge detection is a very basic but important topic in Image processing and Computer Version. To find out the part where the illumination massively varies, techniques of Edge Detection can detect the edges in images, which is very obvious for human's vision system but not for the computer. Therefore, techniques of Edge Detection can provide higher-level information about the images to support higher-level techniques such as Feature detection and Semantic segmentation.

In this part, one of the most intuitive methods of Edge Detection, the Sobel Edge Detection, will be introduced and implemented.

1.1.2. Approaches and Procedures

To find out the areas where the brightness level of pixels varies a lot, Sobel Edge Detection firstly calculated the gradient of pixel values in both directions of X and Y (vertically and horizontally). We can use some specialized filter to get calculate the x-gradient and y-gradient. A simple pair of filters is shown as Figure 1. below.

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

(a)

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

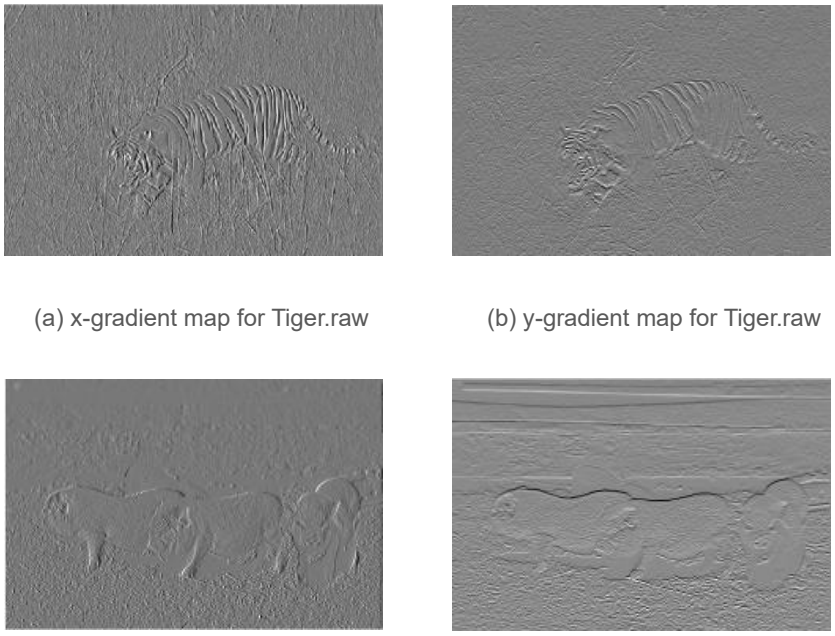
(b)

Figure 1: (a) The filter for x-gradient (b)the filter for y-gradient

Then the geometric mean of each pixel's x-gradient and y-gradient is calculated and we call it the map of gradient **magnitude**. If we want to get a binary edge map, which means the edge pixels are assigned a "255" value and the non-edge pixels are assigned a "0" value, we can choose a threshold to binarize the gradient magnitude map and get an image only contains black and white pixels, where black pixels stand for non-edge areas and white pixels stand for edge areas.

1.1.3. Experimental Results

Step 1: The results of x-gradient map and y-gradient maps for Tiger.raw and Pig.raw are shown in Figure 2. below.

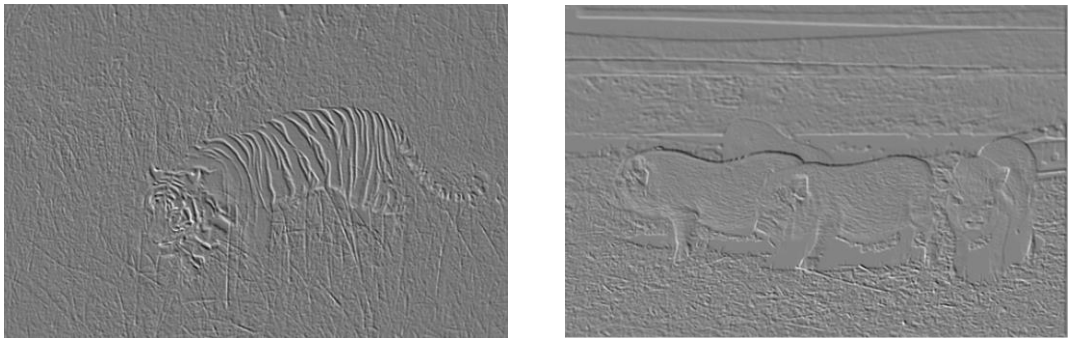


(a) x-gradient map for Tiger.raw (b) y-gradient map for Tiger.raw

(c) x-gradient map for Pig.raw (d) y-gradient map for Pig.raw

Figure 2: x-gradient and y-gradient maps for Tiger.raw and Pig.raw

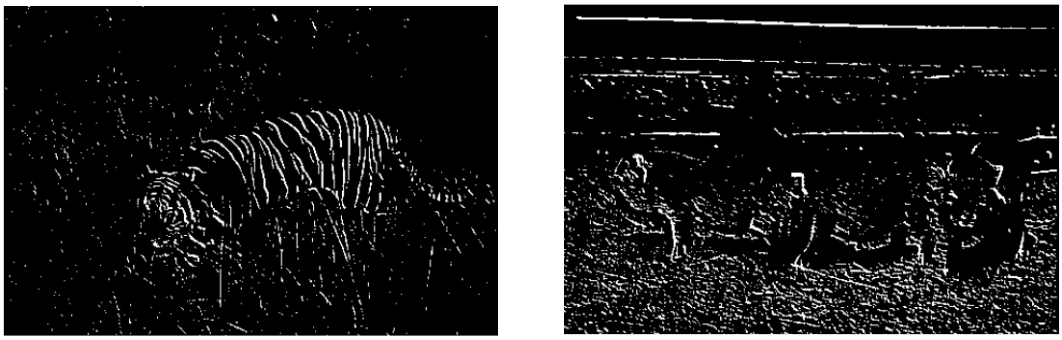
Step 2: The results of magnitude maps of Tiger.raw and Pig.raw are shown in Figure 3 below.



(a) Magnitude map for Tiger.raw (b) Magnitude map for Pig.raw

Figure 3: Magnitude maps for Tiger.raw (a) and Pig.raw (b)

Step 3: The binary edge maps of Tiger.raw and Pig.raw are shown in Figure 4 below. The percentage threshold for Tiger.raw is set as 60.9% and the percentage threshold for Pig.raw is set as 59%.



(a) Edge map for Tiger.raw (b) Edge map for Pig.raw

Figure 4: Binarized edge maps for Tiger.raw (a) and Pig.raw (b)

Although the basic edges are detected by Sobel Edge Detection, but the results are not that good. There are a lot of false alarms, such as the skin patterns of the tiger and the grassy ground around the pigs. Also, there are a lot of missed detection, such as the pigs' backs and the tiger's face and tail. When tuning the threshold values, it's very difficult to optimize the trade-off. If I want to remove more redundant details, some of the edges are also removed, and vice versa. It's fair enough to say that Sobel Edge Detection is the simplest method and that it can only obtain very limited results.

1.1.4. Non-programming Questions

All of the questions HAVE been answered above.

1.2. Part b: Canny Edge Detector

1.2.1. Abstract and Motivation

From the Edge Detection results above, the Sobel Edge Detector and some other very traditional edge detectors can detect pixels in areas of brightness discontinuities. But these when marking out all of these pixels, the whole contours are discrete. However, the real contours, which are actually the real meaningful information about the image, are continuous and connected to each other. Therefore, a more advanced detector, the Canny Edge Detector, who consider the connectivity and continuity of the contours, will be introduced in this part.

1.2.2. Approaches and procedures

The procedures of Canny Edge Detector is divided into three steps:

- Gaussian filtering.
- Non-maximum Suppression.
- Hysteresis Thresholding.

Step 1: Gaussian Filtering: This step is as same as the traditional edge detectors. It applies a Gaussian filter to the image and get the magnitude map to remove the noise and to mark the areas

where the brightness massively changes. Both the gradient magnitude values and gradient direction values (be rounded to simply 4 directions) for each pixel will be recorded.

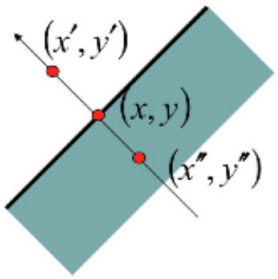


Figure 5: Non-maximum Suppression Operation

Step 2: Non-maximum Suppression: When getting the gradient magnitude and gradient directions for each pixel as mentioned above, the very next step is to apply a non-maximum suppression to every pixel. For every pixel, the gradient magnitude value will be compared to its most adjacent two neighbors, as shown in Figure 5. And the two neighbors are picked according to this central pixel's gradient direction value. For example, if the gradient direction of pixel (x, y) is 135°, the neighboring pixels are picked from its upper-left corner and right-down corner, the pixel (x', y') and (x'', y'').

After selecting the neighbors, the Non-maximum Suppression operation will compare these three pixels' gradient magnitude values to preserve the maximum one and suppress the other lower ones. The algorithm is as below:

$$m(x,y) = \begin{cases} m(x,y), & if\ m(x,y) > m(x',y')\ and\ m(x,y) > m(x'',y'') \\ 0, & otherwise \end{cases}$$

After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image.

Step 3: Hysteresis Thresholding: Even the Non-maximum Suppression can provide an accurate representation of contours of the image, there will still be some unsatisfying parts caused by noises and color variation. To get a more accurate contour of the image, the Canny Edge Detector uses double threshold to detect the real edges and the weak edges. If the pixel is above the higher threshold, the pixel value gradient magnitude is preserved. And if the pixel is between the upper bound threshold and the lower bound threshold, the pixel is representing a weak edge. The pixels below the lower bound threshold will all be suppressed. Whether to preserve the weak edges will be decided by their connectivity with other strong edges.

1.2.3. Experimental results

After applying the Canny Edge Detector on MATLAB using function edge(…) with parameter "canny" in the Image processing toolbox, the detection results are shown as Figure 6 below. The two thresholds [Lower bound, Upper bound] set for Tiger.raw are **[0.16, 0.42]**, and the two thresholds set for Pig.raw are **[0.11, 0.51]**.

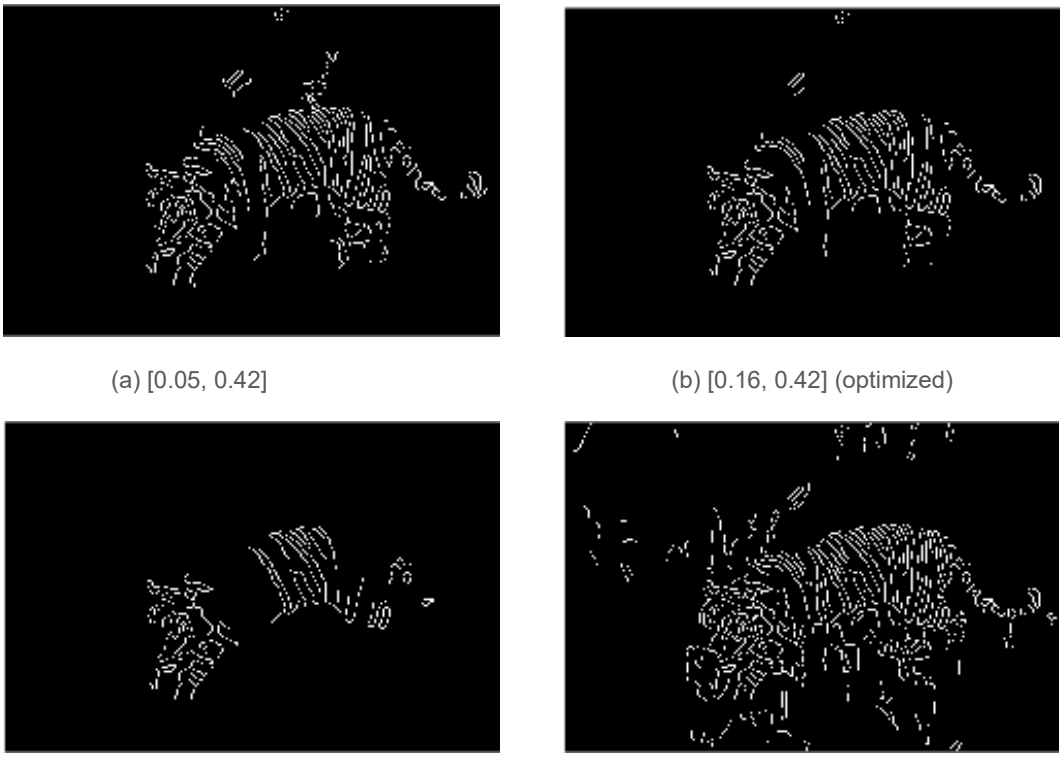
(a) Canny detection for Tiger.raw (b) Canny detection for Pig.raw

Figure 6: Canny Edge Detector results for Tiger.raw (a) and Pig.raw (b)

From the results above, Canny Edge Detector gets very clear and continuous contours for both the images instead of some discrete scatter points of the edge pixels. After selecting the two threshold values distinctively for each image, the noises and color variation parts are effectively removed and the true edges of the images are preserved well.

1.2.4. Discussions

In this section, the threshold values selection and its influence will be discussed. The interface for these two thresholds in MATLAB function edge(…) is defined as a parameter vector filled in a two-value vector [Lower bound, Upper bound]. The Lower bound value should not be bigger than the Upper bound value.



(a) [0.05, 0.42] (b) [0.16, 0.42] (optimized)

(c) [0.16, 0.59] (b) [0.16, 0.25]

Figure 7: Canny Edge Detector results with different threshold parameters [Lower bound, Upper bound]

In Figure 7, several threshold values are used in the Canny Edge Detector and the results are displayed. If the lower bound value is too low, as shown in Figure 7(a), the whole detection results will not be greatly influenced compared to Figure 7(b) which can be the optimized setting for this specific image, but some parts of "false edges" will be counted into the contours, like the areas above the tiger's back. If the upper bound is too large, as shown in Figure 7(c), on the other side, some of the "real edges" will be missed, such as the tiger's chest and tail. And if the upper bound is too small, as shown in Figure 7(d), many of the noises and color variations will be counted into the contours, such as the grassy ground texture, since the Upper bound threshold is the main filter to remove these disturbances.

In conclusion, the two threshold values in Canny Edge Detector can both influence the final results in their own ways. The Upper bound value can determine how much of the noises and color variation disturbances are removed, while the lower bound value can determine how much of the "weak edges" are preserved. For each different scenario and each different image, these two parameters should be selected accordingly and carefully.

1.2.5. Non-programming questions

Q1 and Q2 are answered in section 1.2.2.

Q3 is answered in section 1.2.4.

1.3. Part c: Structured Edge

1.3.1. Abstract and motivation

All the methods introduced above is assumption based, which means that we assume the features of edges ahead and find out the areas that match those features. However, this kind of methods can make a lot of mistakes since many edges are not only depending on the brightness variation but also depending on the humans' understanding. Therefore, the learning-based algorithms are introduced to deal with this situation. The most famous learning-based edge detection methods are Sketch Token (ST) and Structured Edge (SE). The SE detector is the enhanced version of the SE detector, and it will be introduced and tested in this part.

1.3.2. Approaches and procedures

The whole procedure for SE detection is as shown in Figure 8. Each step will be specified below.

- **Step 1. Sampling:** To train the model, a large number of image samples should be collected. The sample images should contain clear edges areas and also non-edge areas. After obtaining the sample images, the images are partitioned into same-sized patches (e.g. 35*35 pxls patches). And all the patches are labeled to match a certain type of edges. This is the preparation for the training data set.

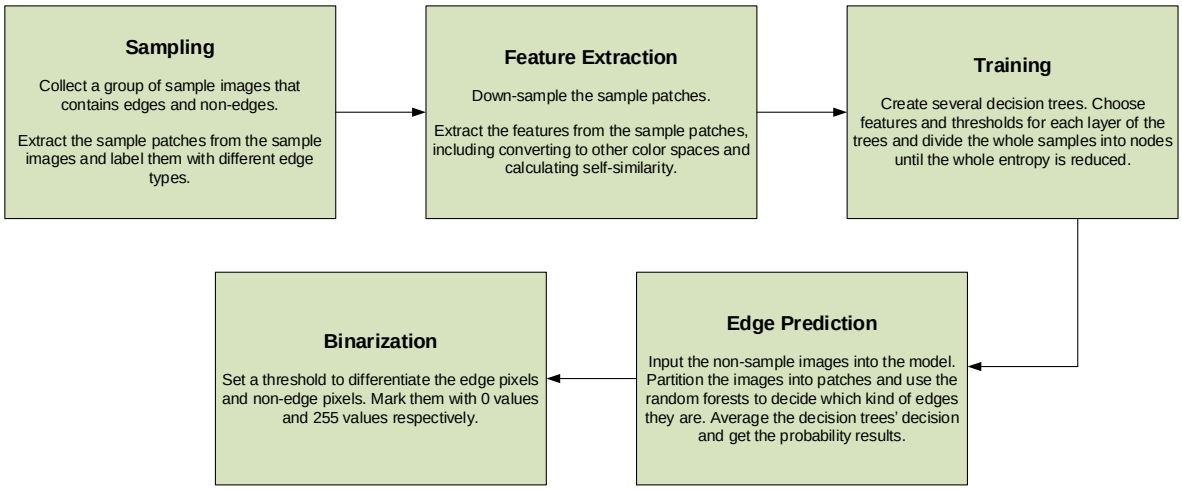


Figure 8: Flowchart for SE detection procedure

- **Step 2. Feature Extraction:** Since every layer of decision tree is separated according to different features, the features of samples should be extracted before training. Each sample patch can be converted to different color spaces, like CIE-LUV space, and it can also be converted to magnitude map and oriented magnitude maps. Each pixel of these different maps can be a feature for a single sample patch. Besides, the sample patches can also be down-sample to smaller size patches and the variation level of each pair of pixels can also be a feature. After extracting all these features, each single patch can have more than 20,000 features.

- **Step 3. Training:** The main part of the SE detector is the Random Forest, which is made up of a certain number of decision tree. Every decision tree is a binary tree and the whole sample patches are separated into two groups level by level according to the selected features and the corresponding thresholds. Every leaf node of each decision tree should contain all the sample patches that contain the same kind of edges. The decision tree numbers and depths are setup ahead of time and part of samples are used to validate the accuracy of the training process. After the validation accuracy is high enough, the training process is finished.

- **Step 4. Edge Prediction:** After the training, the model can be used to detect the edge. When obtaining a new image, the image is firstly be portioned into patches that have the same size as the training data set. Then every patch is input to the random forest model and each decision tree can obtain a classification result, which stands for a certain type of edge. The probability of each type of edge can be calculated from the trees' results and the highest probability is the final results for this single patch. After all the patches get a probability result, the whole probability edge map is obtained.

- **Step 5. Binarization:** In some cases, the probability edge map might not be needed but the binary edge map is needed instead. We can set a threshold to classify the edge pixels and non-edge pixels. If the pixel is an edge, the pixel is assigned a 255 value. Otherwise, the pixel is set a 0 value.

After all the steps above, the edges of an image should be extracted and the results are fully based on samples and training.

1.3.3. Experimental results

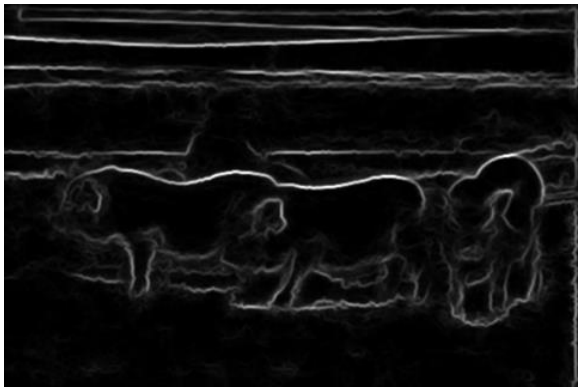
The SE edge detection results are shown as Figure 9 below. The parameters for detection model is set as below. The functions of each parameter has been specified below, and the parameters set below can obtain a high accuracy while the speed of calculation is also guaranteed.

```
model.opts.multiscale=0;      % use multiscale input image to obtain accuary
model.opts.sharpen=2;         % sharpen the output results
model.opts.nTreesEval=4;      % for top speed set nTreesEval=1
model.opts.nThreads=4;        % max number threads for evaluation
model.opts.nms=0;             % set to true to enable non-maximum suppression
```

Besides, the threshold value of both image is set as 18% of the max value to obtain the binarized edge map.

(a) Probability edge map for Tiger.raw

(b) Binary edge map for Tiger.raw



(c) Probability edge map for Pig.raw

(d) Binary edge map for Pig.raw

Figure 9: SE detection results for Tiger.raw and Pig.raw

As the results shown in Figure 9, the SE detector can obtain a very good result. The edges of both images are detected accurately while the textures and grassy ground are removed perfectly. Since, the SE detector is training-based, it can handle well with the weak edges and the false alarm parts.

1.3.4. Non-programming questions

Q1: Please digest the SE detection algorithm. Summarize it with a flow chart and explain it in your own words.

A1: This question has been answered in section 1.3.2 Approaches and procedures

Q2: Explain the process of decision tree construction and the principle of the RF classifier.

A2: Random Forest (RF) model is a typical kind of ensemble training, which means it use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In RF model, it uses multiple decision trees to do the classification and then obtain the results synthesizing the results from all the decision trees. If the classification is discrete, the RF model uses the mode from all the results. And if the classification is regression, the RF model uses mean value of all the results to calculate the final result.

As for the decision tree, every decision tree is a binary tree. The root node of the decision tree contain all the samples, and with every classification, the whole simples from the root node are separated into two groups, which contains different sample that have different features according to the features selected for this level. As the decision tree grows deeper and larger, the classification is finer and finer and finally the whole samples are classified into the number of classes as we expected. Since every tree can get different results as the selected features and growing orders are all different, the RF model uses the mode of the results from all the trees and it preserve the accuracy and prevent the overfitting flaw.

Mode specifications of the RF model used in the SE detector are in section 1.3.2 Approaches and procedures.

1.4. Performance Evaluation

1.4.1. Abstract and motivation

To evaluate an edge detector's results, simply observing is definitely not enough. Usually, the edge detector's results should be compared to the edge maps marked by real human beings, which are called the ground truth. There are two typical merits to evaluate the edge detector's results when compared to the ground truth, which are Precision and Recall. The Precision value can indicate how many edge pixels are real edge pixels among all the edge pixels that are picked by the edge detector, and the Recall value can indicate how many edge pixels are recognized successfully from all the real edge pixels. To get a better result, Precision and Recall values should all be higher, but these two values are mutually restrained, which means we can only find the optimized trade-off between these two indicators. The F-measure is a formula of the relation between Precision and Recall. To obtain higher Precision and Recall, we can simply maximize the F-measure. The specified calculation will be introduced in next section.

1.4.2. Approaches and procedures

The Precision and Recall are calculated as following:

$$Precision\ P = \frac{True\ positives}{True\ positives + False\ positives}$$

$$Recall\ R = \frac{True\ positives}{True\ positives + False\ negetives}$$

The F-measure is calculated as following:

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)$$

$$F = 2\,\frac{P \times R}{P + R}$$

In this part, both Tiger image and Pig image are provided 5 ground truth images respectively and the evaluation merits mentioned above will be implemented and be applied to the edge maps obtained from the detectors in previous sections. Also, several threshold values will be set to the SE detector to find out the influence to the F-measures when trying different threshold values.

1.4.3. Experimental Results

Step 1. Basic evaluations for different Edge detectors:

The evaluation results are shown in Table 1 below. The Precision and Recall values are separately calculated for each ground truth image and the F-measures are obtained from the Precision and Recall values. The average F-measure for both Tiger image and Pig image are calculated and also the whole F-measure for the detector.

As the results indicate, the F-measure of Sobel detector is the lowest since the Sobel detector is the simplest and the crudest one. The Canny detector gets a little bit higher F-measures scores and the SE detector's F-measure scores is much higher since the SE detector's results are much more like what human beings' decisions.

In regard to different kind of images, the performance for each edge detector is also different. Since the edges of the Tiger image is easier to detect, the simplest Sobel detector's F-measure scores for Tiger image is much higher than the those for Pig image, which means the Sobel detector has less robust ability. However, the F-measure scores of Canny detector and SE detector in regard to these two different pictures are quite similar to each other, which means these two kinds of detectors have already got some adaptability under different situations.

Table 1. Evaluation results for Sobel detector, Canny detector and SE detector

Edge detector	Ground Truth Index		Precision	Recall	F-measure	Average F-measure	
Sobel Detector	Tiger Image	1	0.013343	0.065078	0.022146	0.056794	0.053953
		2	0.014471	0.066551	0.023773		
		3	0.015599	0.063895	0.025076		
		4	0.183988	0.191136	0.187494		
		5	0.01635	0.057731	0.025483		
	Pig Image	1	0.025297	0.150314	0.043306	0.051111	
		2	0.02489	0.13679	0.042117		
		3	0.039125	0.134283	0.060595		
		4	0.034163	0.093292	0.050012		
		5	0.03823	0.134401	0.059528		
Canny Detector	Tiger Image	1	0.038773	0.140238	0.06075	0.085195	0.089584
		2	0.034465	0.117545	0.053302		
		3	0.03852	0.117013	0.05796		
		4	0.23112	0.178055	0.201147		
		5	0.036493	0.095554	0.052815		
	Pig Image	1	0.077031	0.106332	0.08934	0.093973	
		2	0.111695	0.142602	0.12527		
		3	0.102241	0.081519	0.090711		
		4	0.085434	0.054198	0.066322		
		5	0.109244	0.089219	0.098221		
SE Detector	Tiger Image	1	0.068971	0.384051	0.116941	0.145796	0.146719
		2	0.077695	0.407952	0.130531		
		3	0.073086	0.341801	0.120423		
		4	0.23358	0.27704	0.253461		
		5	0.06716	0.270737	0.107623		
	Pig Image	1	0.061627	0.464959	0.10883	0.147643	
		2	0.066432	0.463567	0.11621		
		3	0.103652	0.451703	0.168612		
		4	0.134914	0.467792	0.209427		
		5	0.082703	0.369174	0.135134		

Step 2. Evaluations for edge detectors with different threshold values:

As introduced above, the real edge maps are obtained from the probability maps with different threshold values set to binarize the probability maps. To study the influence that different threshold values might bring to the final results, a range of threshold values in [0.01:0.01:1] are set to the SE detectors and the F-measure scores are calculated and plotted corresponding to the threshold values. The threshold values are in percentage manner, which means the real thresholds are calculated as

Percentage * (the max probability value) in the probability map. The results are shown in Figure 10 below.

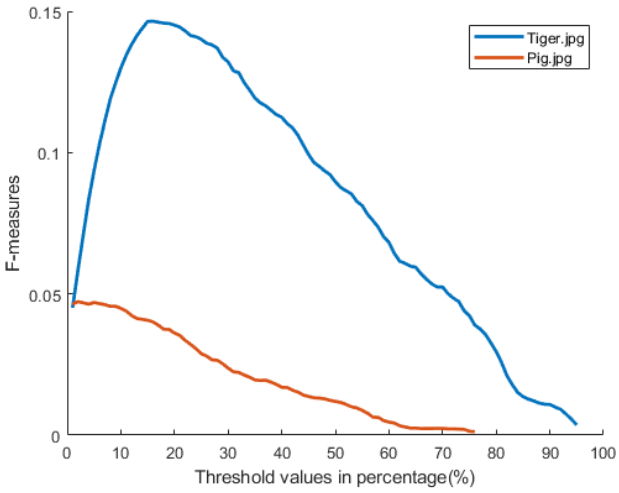


Figure 10: SE detector's F-measure scores under different threshold values

As indicated in Figure 10, for the Tiger image, the maximized F-measure occurs when the Threshold value is set to be 0.16 and the maximized F-measure score is 0.1465. If the threshold is too high or too low, the F-measure scores will decrease. However, for the Pig image, the curve is monotonically decrement, which means that the maximized F-measure score occurs when the Threshold value is set to be 0.01 or lower.

1.4.4. Non-programming questions

The first two questions in the HOMEWORK REQUIREMENTS have been answered in section 1.4.3.

Q3: The F measure is image dependent. Which image is easier to a get high F measure – Tiger or Pig?

Answer: According to Table 1 above in section 1.4.3, the Pig image are more likely to get a higher F-measure scores, because the Canny detector and the SE detector both get higher F-measure scores in Pig image than in Tiger image. The most possible reason for this result can be that the skin and pattern on the tiger are very deceptive, which the algorithms can recognize to be edges because the brightness of these areas vary a lot, but the human beings may not admit them to be the edges because it does not accord to our universal common sense if we do. The evidence to support this deduction is that if we pay attention to the ground truth NO.4 for Tiger image, we can find that the F-measure scores are much higher than the other ones', because only this ground truth image marks all the patterns and skin stripes on the tiger and this is exactly what all the edge maps appear to be. Therefore, the Pig image is much easier to get a higher F-measure score.

Q4: Discuss the rationale behind the F measure definition. Is it possible to get a high F measure if precision is significantly higher than recall, or vice versa? If the sum of precision and recall is a constant, show that the F measure reaches the maximum when precision is equal to recall.

Answer: The F-measure is the harmonic mean of both the Precision and Recall values. The harmonic mean formular is given below with n members x_i, and it pays more attention to the lowest member in the group while doing the averaging. In this case, the F-measure uses harmonic mean not only because the F-measure is higher when Precision and Recall are both higher, but also to prevent any of these two indicators gets too low and to balance the trade-off between these two indicators.

$$H_n = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}} = \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}}$$

Assume the sum of P and R is a constant, that is, $P + R = C$, F-measure formula can be converted as following:

$$F = 2 \cdot \frac{P \times R}{P + R} = 2 \cdot \frac{P(C - P)}{C} = 2 \cdot \frac{-P^2 + CP}{C}$$

The term $(-P^2 + CP)$ reaches maximum when $P = C/2$, that is, $P = R$. Therefore, if P is significantly greater than R or R is significantly greater than P, which means P is much greater than $C/2$ or P is much less than $C/2$, the term $(-P^2 + CP)$ will decrease and the F-measure can't be higher but lower. Only when Precision is equal to Recall can we get the maximized F-measure on condition that the sum of Precision and Recall is a constant.

II. Problem 2: Digital Half-toning

2.1. Part a: Dithering

2.1.1. Abstract and motivation

Halftone is the reprographic technique that simulates continuous-tone imagery through the use of pixels that only have binary values. Since the printers and the fax machines can only render a binary value on a certain pixel, which means that it can either put some ink on the pixel to make it 255 or not put some ink to make it 0. The most intuitive method to halftone is set a threshold and binarize the image. However, simply setting a threshold can hardly get good results and the results usually do not conform to people's perceptions. Some advanced halftoning methods will be introduced and implemented in this part.

2.1.2. Approaches and procedures

In this part, three methods of halftoning will be implemented: Fixed thresholding halftoning, random thresholding halftoning and Dithering.

The first two are the most intuitive ones. The halftoned pixel values are acquired by the formula below, where the original pixel value is denoted as $F_{(i,j)}$, the halftoned pixel value is denoted as $H_{(i,j)}$ and the threshold value is denoted as T. For the fixed thresholding method, T is a fixed value and we usually set it to be 128. For the random thresholding method, T is a random variable in the range of 0 to 255 and we usually set the T to be uniformly distributed.

$$H_{(i,j)} = \begin{cases} 0 & if\ F_{(i,j)} < T \\ 255 & if\ F_{(i,j)} \geq T \end{cases}$$

The other method, dithering, is much fancier than the two methods mentioned above. Firstly, the dithering matrix is chosen and initialized as following:

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

$$I_{2N\,(i,j)} = \begin{bmatrix} 4 \times I_{2\,(i,j)} + 1 & 4 \times I_{2\,(i,j)} + 2 \\ 4 \times I_{2\,(i,j)} + 3 & 4 \times I_{2\,(i,j)} + 0 \end{bmatrix}$$

Then the threshold values are acquired from the dithering matrix by the formula below, where x and y are real pixel coordinates and “mod” operation is getting the remainder. Then the threshold matrix is applied to the image and the halftoned imaged is obtained.

$$Threshold_{(x,y)} = \frac{I_{N\,(x\ mod\ N,\ \ y\ mod\ N)} + 0.5}{N^2} \times 255$$

2.1.3. Experimental results

Figure 11 shows the results obtained by the fixed thresholding halftoning (left) and the random thresholding halftoning (right). The fixed threshold value is set as 128 and the random threshold value is set to be uniformly distributed as $U(0,255)$.

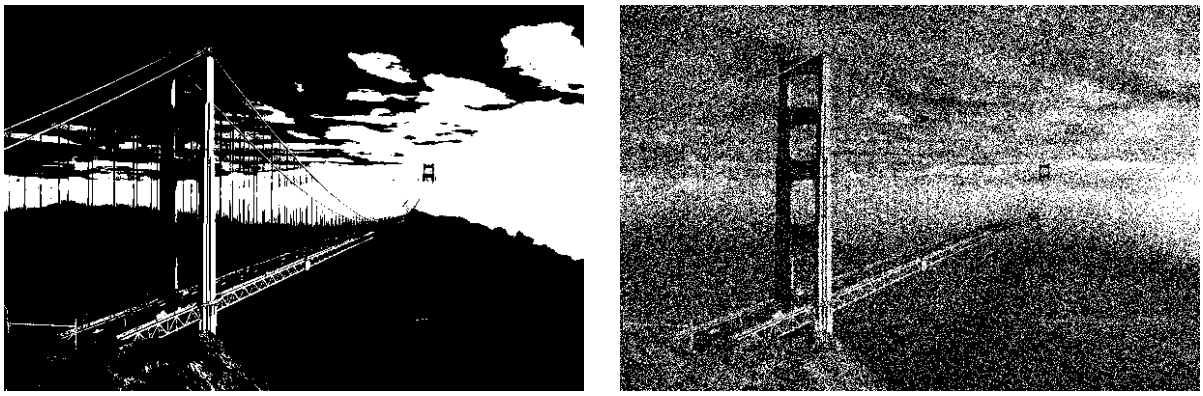


Figure 11: Fixed thresholding halftoning results (left) and random thresholding halftoning results (right)

The dithering matrixes and their halftoning results are shown in Figure 12, 13 and 14 below.

1	2
3	0

Figure 12: I_2 dithering matrix and its halftoning result

21	37	25	41	22	38	26	42
53	5	57	9	54	6	58	10
29	45	17	33	30	46	18	34
61	13	49	1	62	14	50	2
23	39	27	43	20	36	24	40
55	7	59	11	52	4	56	8
31	47	19	35	28	44	16	32
63	15	51	3	60	12	48	0

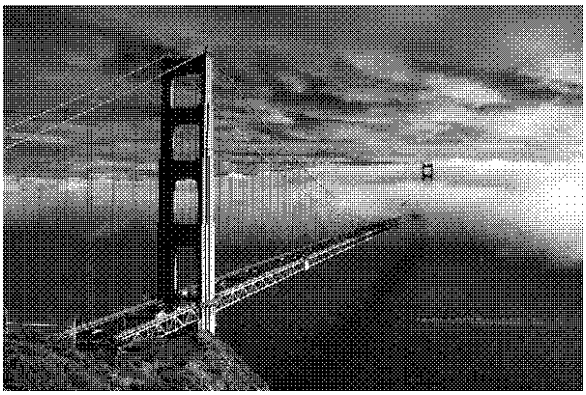
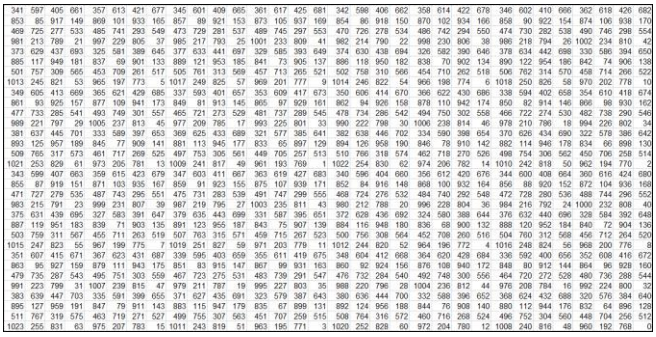


Figure 13: I_8 dithering matrix and its halftoning result



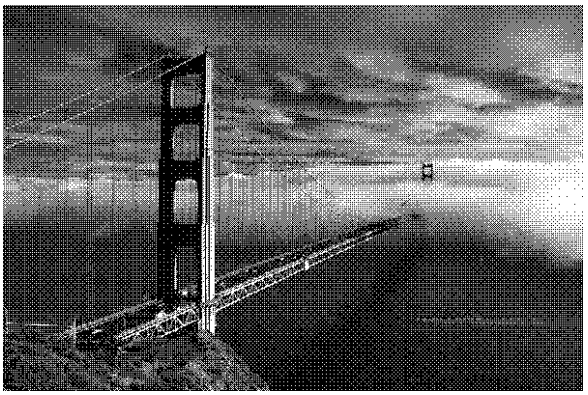


Figure 14: I_{32} dithering matrix and its halftoning result

2.1.4. Discussions

According to Figure 11, the results from fixed thresholding and random thresholding are unsatisfactory. The fixed thresholding loses many details of the image such as the textures and edges, while the random thresholding can preserve some details but the visual effects are quite awful.

When using dithering halftoning, as shown in Figure 12 through 14, the overall effects are much better. The whole image's contents and edges are all recognizable and clear. If using the smallest dithering matrix I_2, as shown in Figure 12, the edges and gradients are sharpened and not that smooth, especially for the textures on the sky and the ocean. However, this situation is largely alleviated if using lager matrixes I_8 and I_{32}, as shown in Figure 13 and Figure 14. The edges and gradients are smoothened and the whole visual effects are much more natural.

However, since the dithering uses formatted matrixes periodically, the halftoned is inevitably covered with some kind of mesh textures. This is dithering's shortcoming and might cause problems under some certain occasions.

2.1.5. Non-programming problems

All the non-programming problems **ARE** answered in previous sections.

2.2. Error diffusion

2.2.1. Abstract and motivation

Error diffusion is another method of halftoning that has totally different train of thought than dithering halftoning. Error diffusion's main idea is to diffuse the errors from the binarized value and the original value to the neighboring pixels that have not been processed yet. Unlike many other halftoning methods, error diffusion is classified as an area operation, because what the algorithm does at one location influences what happens at other locations. Some may clarify that error diffusion has the

tendency to enhance edges in an image and that it can make text in images more readable than in other halftoning techniques.

2.2.2. Approaches and procedures

First of all, the error diffusion matrix is picked and initialized. There are three error diffusion matrixes that will be applied to the image. They are Floyd-Steinberg's error diffusion matrix:

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

Jarvis, Judice and Ninke (JJN)'s error diffusion matrix:

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

And Stucki's error diffusion matrix:

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

When conducting Error diffusion halftoning, the very first step is to slide the matrix kernel on the image following a specific path and to binarize the central pixel value using a fixed threshold. After the binarization, the error between the binarized value and the original value can be calculated:

$$e_{(i,j)} = \tilde{f}(i,j) - b(i,j)$$

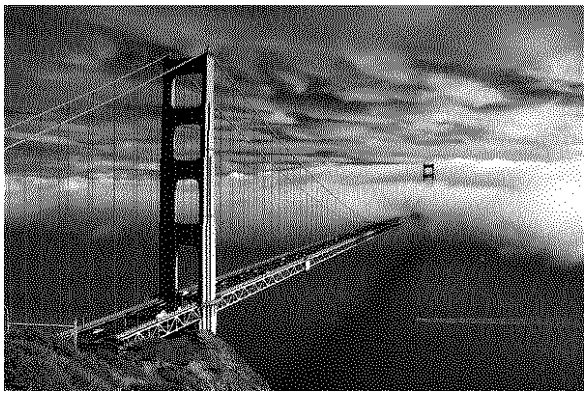
And then the error values are diffused and accumulated on the unprocessed pixels following the central pixel, as indicated in the error diffusion matrixes. And the new pixel values of the following pixels are calculated as the formula below, where m and n are error diffusion matrix $h(m,n)$'s coordinates.

$$\tilde{f}(i+m, j+n) = f(i+m, j+n) + e(i,j) \times h(m,n)$$

The steps above are repeated as the error diffusion matrix is proceeding on the image and all the pixels are gone over.

2.2.3. Experimental results

Figure 15 shows the results of Error diffusion halftoning results using different Error diffusion matrixes among Floyd-Steinberg matrix (a), JJN matrix (b) and Stucki matrix (c).



(a) Floyd-Steinberg matrix's results

(b) JJN matrix's results

(c) Stucki matrix's results

Figure 15: Error diffusion halftoning results with different matrixes

2.2.4. Discussion

As shown in Figure 15, the Error diffusion method with all the three different matrixes halftones the image perfectly. Compared to the dithering method, the results of error diffusion have smoother and more natural edges and gradients. Also, Error diffusion method does not add any periodical pattern to the halftoned image but the whole distribution is more random, which is better than the dithering method. Different error diffusion matrixes can make little difference compared to each other, but the smaller matrix (Floyd-Steinberg matrix) can obtain smoother halftoned results while the larger matrixes (JJN matrix and Stucki matrix) can enhance the edges and obtain more detailed textures. From my prespective, the results from Error diffusion halftoning are preferred.

To get a better result based on Error diffusion method, my idea is to make larger error diffusion matrix, such as a 7-by-7 matrix or larger, and to make the error factors more intense around the center of the matrix, which means the pixels who are nearer to the central pixel get more error diffusion from the central pixel. In this way, the error can be more widely diffused, and the edges can be better preserved. And most importantly, the halftoned image will not be covered with periodical patterns like dithering results because error diffusion operation is pixel-wise but not area-wise. Of course, the Error diffusion matrix should be chosen according to different occasions and requirements.

2.2.5. Non-programming questions

All the questions **HAVE** been answered in previous sections.

III. Problem 3: Color halftoning with Error diffusion

3.1. Separable Error Diffusion

3.1.1. Abstract and motivation

Several methods of halftoning on the one-channel gray scale images have been introduced and implemented in the last Problem section. In this problem section, the halftoning techniques will be expanded to be applied to the color images with three channels standing for RGB, specifically, the Error diffusion halftoning will be expanded.

The most intuitive method to halftone the color image is to process the three channels separately and then combine the results. In this part of the problem, separable error diffusion operation will be implemented and discussed.

3.1.2. Approaches and procedures

Before the separable error diffusion operations, the RGB image is firstly converted to the CMY color space which stands for Cyan, Magenta and Yellow. The conversion is conduct by the following formula (assuming that RGB values range from 0-255):

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = 255 - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Then the error diffusion operations is conduct on the CMY channels separately as the same as that in section 2.2.2, with Floyd-Steinberg's error diffusion matrix. After the error diffusion halftoning, the halftoned image is converted back to RGB color space using the following formula. And the halftoned color imaged is obtained.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = 255 - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

3.1.3. Experimental results

The separable error diffusion result is shown in Figure 16 below. The effects of the color halftoning is quite similar to the grayscale halftoning if using the error diffusion method. If zoomed in, it can be noticed that there are actually 8 types of colors that exist on the color image, who stand for 8 combinations of R G B values at 0 and 255.

Figure 16: Color halftoning results using separable error diffusion method

3.1.4. Discussions

Although the overall visual effects of separable error diffusion halftoning is quite satisfying, some flaws can still occur on this method. The main shortcoming is that the separable operations can bring color distortions to the halftoned image. If zooming in the image, there are some areas have color distortions, especially for the areas of edges where the pixels' colors vary a lot from each other. Since the error diffusion is operated completely independently on each channel, the error corrections are all from the central pixels of the single channel that is being processed. For the areas that have less color variation, this can cause few problems because the grayscale values in three channels are almost the same. But for those areas that have heavy colors or massive color variations, the error diffusion differences in three channels can influence the final results largely. Therefore, the color distortions are brought to the halftoned image and this is somehow inevitable if we choose to process the three channels separately.

3.1.5. Non-programming questions

All the questions **HAVE** been answered in previous sections.

3.2. MBVQ-based Error diffusion

3.2.1. Abstract and motivation

As indicated in section 3.1.5 above, the separable error diffusion halftoning is quite close to get the perfect results, but processing the three channels separately can still bring some problems inevitably. There are some methods to halftone the three channels jointly and the error diffusion based on MBVQ (Minimal Brightness Variation Quadruples) can work well among all. In this part, the MBVQ-based error diffusion halftoning method will be introduced and implemented. And the results will be shown and compared to the separable error diffusion.

3.2.2. Approaches and procedures

The MBVQ-based error diffusion is similar to the normal error diffusion in most the parts. The different and the most important part in it is the thresholding step, which uses quadruples in the color spaces and decides the closest vertex a certain pixel belongs to.

The color space formed by three axes are partitioned into 6 quadruples as following Figure 17 indicates and the first step in MBVQ step is to decide which quadruple the central pixel is in. Then the central pixel is assigned the RGB value which the closest vertex in the quadruple has. For example, if the central pixel is in the quadruple “CMYW” and the closest vertex is Y, then the RGB value should be converted to (255, 255,0), which is actually “yellow” color’s RGB value.

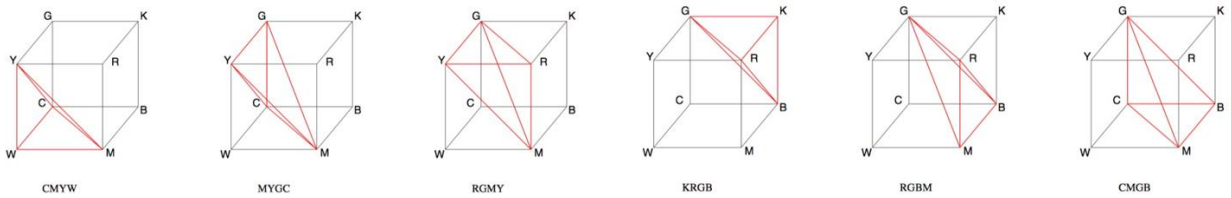


Figure 17: MBVQ’s names and definition

After the MBVQ operations, the normal error diffusion is then applied to the pixels. The error between the central pixel and the converted pixel will be calculated and be diffused to the neighboring pixels using Floyd-Steinberg’s matrix. The diffuse errors are accumulated on each unprocessed pixel and the error diffusion kernel is sliding on the whole image until all the pixels are gone over.

3.2.3. Experimental results

The MBVQ-based error diffusion result is shown in Figure 18 below.

Figure 18: MBVQ’-based error diffusion halftoning result

3.2.4. Discussions

Compared to the result from separable error diffusion halftoning, the result from MBVQ-based halftoning has less color distortions, especially in the areas that have pure colors. As shown in the Figure 19 below, in the white areas marked by the red rectangles, the result from separable halftoning has some dark pixels (blue and black ones) while the result from the MBVQ-based halftoning nearly has no distortion pixels. Most pixels in the result obtained by the MBVQ-based halftoning conform to the overall level of brightness of their surrounding areas, which means if the whole area is dark, all of the pixels in this area will have lower brightness color like black and blue.

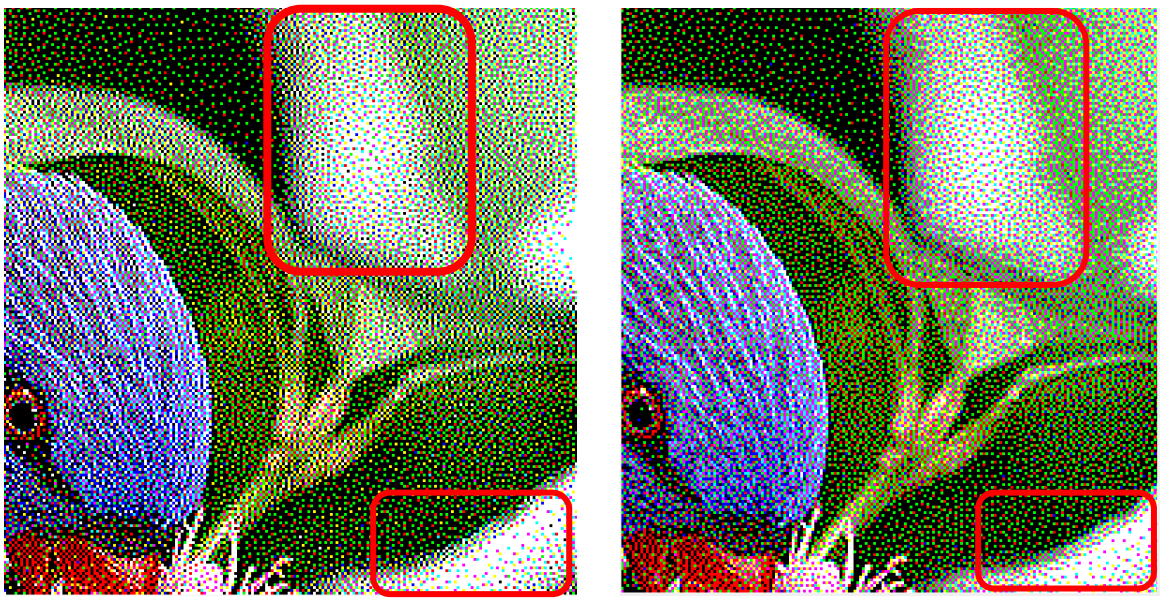


Figure 19: Detailing compare of Separable error diffusion result (left) and MBVQ'-based error diffusion halftoning result (right)

3.2.5. Non-programming questions

Q1: Describe the key ideas on which the MBVQ-based Error diffusion method is established and give reasons why this method can overcome the shortcoming of the separable error diffusion method.

Answer: The key ideas on which the MBVQ-based error diffusion method has been elaborated in section 3.2.2. MBVQ-based error diffusion can overcome the shortcoming of the separable error diffusion because MBVQ-based error diffusion method quantitates the pixel values combining all the three channel and makes the decision together while the separable error diffusion process the three channels completely individually. If some channel's color varies more than the other channels in some areas, the variation can't be diffused to the other channels. But the MBVQ-based error diffusion can solve this problem because when the error from three channels will all be calculated and diffused to the neighboring pixels and the color is equalized.

The other questions are answered in previous sections.