EE 569 Digital Image Processing

Homework2

Problem 1. Edge Detection

1.1 Abstract and Motivation

Edge detection is a important method in image processing, it can get the edge of the object in the figure. For traditional edge detection, it works by detecting continuity of pixels' value. If there exists sudden change, it can be considered as an edge in high probability. For data driven edge detection, it build machine learning-based model first, and use the model to generate the edge.
In this report, there are three different implementations of edge detection. They are sobel edge detection, Canny edge detection and structured edge detection. First two edge detection are traditional edge detection, the last one is data driven edge detection.

1.2 Sobel Edge Detection

1.2.1 Approach and Procedure

First of all, the figure should be transferred to gray scale. After this operation, we can start apply the sobel edge detection to the figure. For sobel edge detection, we need to calculate the x-gradient and y-gradient by using the filter shown as below.

$$x-gradient\text{ - }filter=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$

$$y-gradient-filter=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

After using filter to generate the x-gradient matrix and y-gradient matrix, I combine them to get the edge map of the whole figure.

1.2.2 Experiment result

(1)
Figure 1 and Figure 2 shows the x-gradient and y-gradient of Gallery.jpg and Dogs.jpg respectively.

Figure1.1 X-gradient of Gallery.jpg

Figure1.2 y-gradient of Gallery.jpg

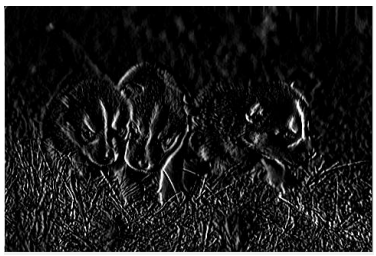


Figure2.1 x-gradient of Dogs.jpg

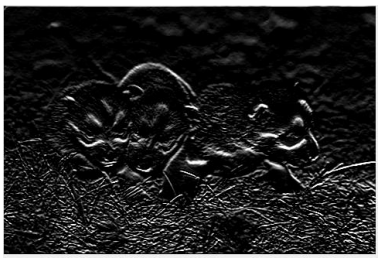


Figure2.2 y-gradient of Dogs.jpg

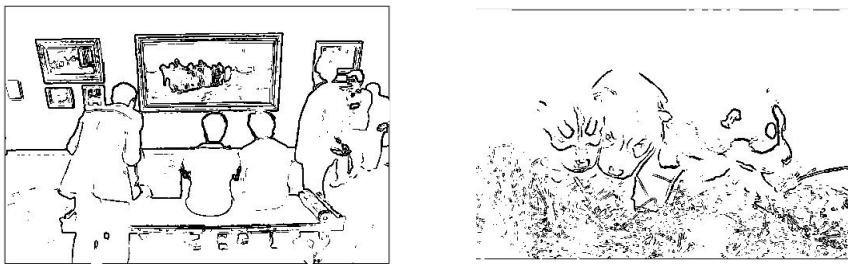


Figure3 edge map of Gallery.jpg and Dogs.jpg

1.2.3 Discussion

To get the best edge map, I tried p in range of (0.1,0.2) which represent threshold in range of 90% to 80%. Personally speaking, 80% has best performance for Gallery.jpg. For Dog.jpg, threshold equals to 85% has best visual performance.

1.3 Canny Edge Detection

1.3.1 Approach and Procedure

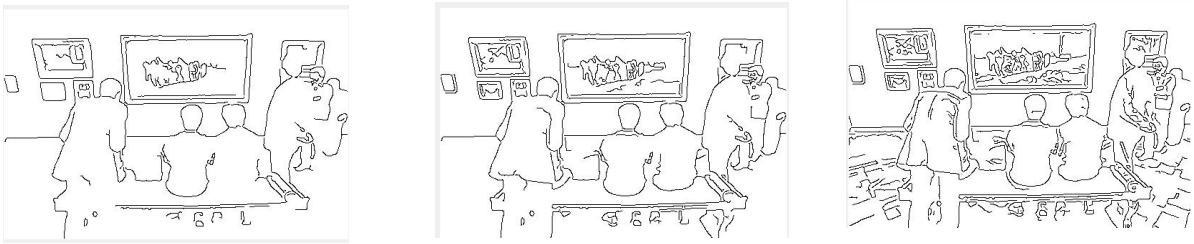
(1) In Canny Edge Detection, non-maximum suppression is a great technique to alleviate the blur boundary. First of all, each pixel would get it's gradient direction approximate value in range of [0,45,90,135,180,225,270,315]. Which represents up, upper right, right, lower right, down, lower left, left and upper left. Then compared the gradient value of pixel with the pixels in the positive and negative gradient direction. If the gradient value is the maximum, we keep it. Otherwise, we suppress it(set it to zero).

(2)

In Canny Edge Detection, we also set for two threshold. If the pixel value is higher than higher threshold, we would keep it and define it as strong edge. If the pixel value is smaller than lower threshold, it would never be considered as edge. For those pixels between higher threshold and lower threshold, it would be considered as weak edge. To decide whether it can be keep or not, we need to search the pixels around it. If there exists a strong boundary connect to weak boundary, it could considered as part of strong edge. Otherwise, when there is only pixels smaller than lower threshold connect to it, it won't be classify as edge.

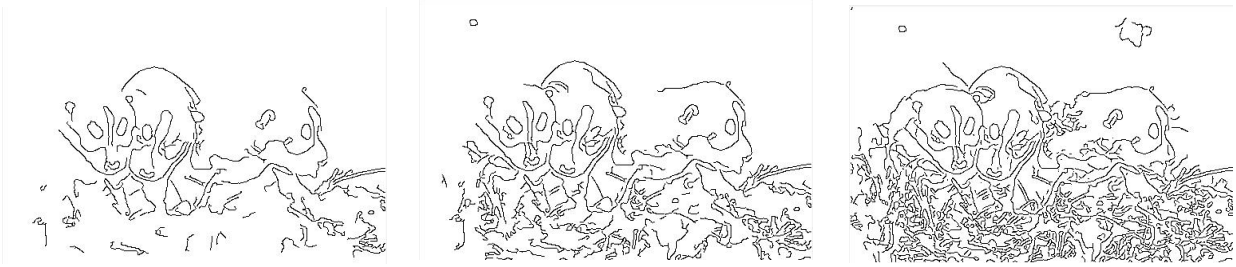
1.3.2 Experiment Results

In this report, we use a fixed ratio of high threshold and low threshold, which is 0.4. Then we choose 0.3, 0.2 and 0.1 as high threshold respectively for Gallery.jpg.For Dogs.jpg, we choose 0.4,0.3 and 0.2as high threshold. The result of Gallery.jpg and Dogs.jpg is shown in Figure4 and Figure5.



(a)High T=0.3,low T =0.12 *(b)High T=0.2,low T =0.08* *(c)High T=0.1,low T=0.04*

Figure4. Different threshold for Gallery.jpg



(a)High T=0.4, low T = 0.16 *(b)High T=0.3, low T = 0.12* *(c)High T=0.2, low T = 0.08*

Figure5. Different threshold for Dogs.jpg

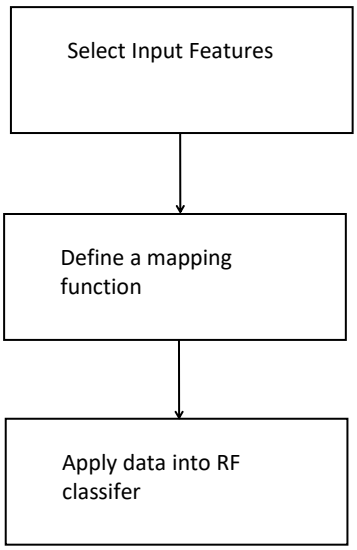
1.3.3 Discussion

First of all, the lower threshold can't be too high. Or the result is shown as figure4(a). The most of edge won't be detected in this kind of situation. However, if the threshold is set to a really low number, some unimportant texture would shown in the result. In general cases, the ratio between higher threshold and lower threshold is 0.4. In my point of view, high threshold for 0.2 and low threshold for 0.08 is a good choice for Gallery.jpg. For Dogs.jpg, the best choice for high threshold and low threshold is 0.4 and 0.16 respectively.

1.4 Structured Edge

1.4.1 Approach and Procedure

(1)



Basically, structured edge is based on the random forest algorithm. In this method, the goal is to transfer a pixels matrix to a matrix contains information of whether a pixel contains edges. First of all, input features need some operation. Instead of using values in R,G,B channels directly, we compute 3 color channels in CIE-LUV color space and add filter triangle filter to get candidate features as our input. For mapping function, we define a alternate mapping function. To be more specific, the mapping function shows the result of whether two given pixels in figure are belongs to same segment. After an edge is defining, whether two pixels are in the same segment is also decided. Also it has only 256*255 dimension, so it is implementable. Finally, the random forest model

(2)

Before introduce the RF classifier, I would like to introduce the decision tree first. When we get a batch of data, getting the features is the first step. By comparing the information entropy of the whole tree and original information entropy, information gain can be calculated. The feature which can get the highest gain is considered as root of the tree. The highest gain means the fastest information entropy dropping rate. Using recursion, we can make sure which feature can be used as root of sub-trees. When all the features are used or the entropy decrease to zero, process of construction is end. The formula of information entropy and information gain is shown as below.

$$entropy(D) = -\sum_{k=1}^{y} p_k \log_2 p_k$$

$$Gain(D,a) = entropy(D) - \sum_{sub=1}^{sub} \frac{|D_{sub}|}{|D|} entropy(D_{sub})$$

Random forest classifier is based on bagging, so I would like to give a rough idea of bagging first.Bagging is based on bootstrap aggregation. First step of bagging is sampling for n samples(can be repetitive) from the sample set. Secondly, construct classifier for n samples based on all given features. Then, Repeat the same process for m times, we can get m different classifiers. Finally, use m classifiers classify data. Based on the polling result of m classifiers, the final result can be generated. Random forest also use the bootstrap aggregation to randomly select samples from sample set with replacement. For feature selection part, we don't use all the feature we have to construct one decision tree. Instead, we randomly choose k features to construct the tree. Repeat the same process for m times, we get m decision tree,which form our random forest.

(3)

For the given source code, there has 5 parameters should be set.

(1) Multiscale

I set the parameter for 1 here. As mentioned in annotation, for top accuracy, we need to set the Multiscale to 1. To be more specific, program would run structured edge detection on three versions and get the average value.

(2) Sharpen

For this parameter, the function is to get a sharper edge. Although we have non-maximum suppression, we can still use sharpen to get a even better edge figure. The maximum value of sharpen is 2. As for best result, we set sharpen as 2.

(3) nTreesEval

It's a parameter to define the number of decision trees in our random forest tree. As we known, the larger the forest is, the better the performance is. Larger forest consist more result of decision trees, so that we could get a better performance. Here we set it to 10.

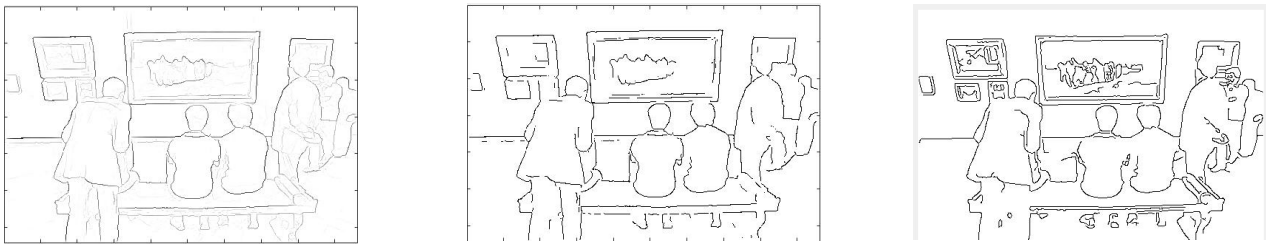
(4) nThreads

For this parameter, personally I define it as a time-costing related parameter. It defines the maximum number of threads for evaluation. So I just use the default value 4 to achieve a high speed evaluation.

(5) Non-maximal suppression(nms)

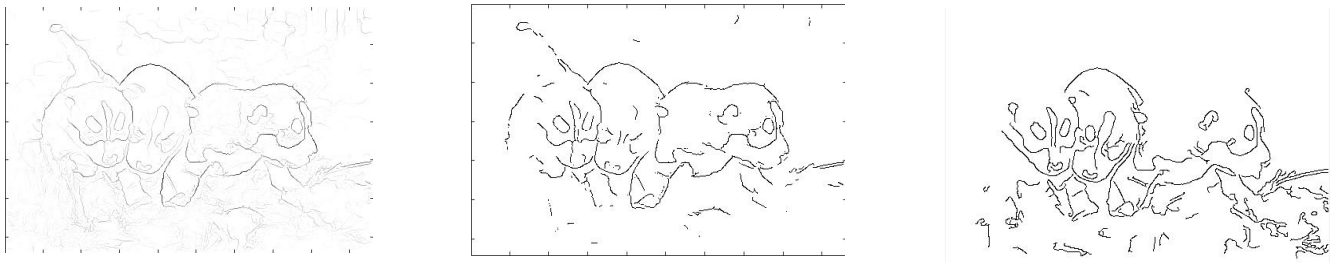
Nms is also set to true here to activate the non-maximal suppression process. As we mentioned in the early section of the report, non-maximal suppression can alleviate the blur boundary. To get a better performance, we set it to 1 here.

1.4.2 Experimental Result



6.1 Probability edge map(inversed) *6.2 Binary edge map(with p>0.2)* *6.3 result for Canny*

Figure6.result of Gallery.jpg using SE compared with Canny detector



7.1 Probability edge map(inversed) *7.2 Binary edge map(with p>0.15)* *7.3 result for Canny*

Figure7.result of Dogs.jpg using SE compared with Canny detector

1.4.3 Discussion

For the probability threshold, I choose 0.2 and 0.15 for 'Gallery.jpg' and Dogs.jpfg respectively. Compared with Canny detector, structured detector can only show the shape or outline of object. For canny detector, it can focus on the relationship between pixels and surroundings, so that Canny detector can be interfered by noise. For example, in Gallery.jpg Canny detector can detect the texture of floor and details of picture on the wall, which is actually not the edges we truly want. For

Structured edges detector, it's a data driven algorithm. So that it can ignore most of noise. However, it can easily cut down some edges.

1.5 Performance Evaluation

1.5.1 Experimental Results

(1)

	Gallery.jpg(binarized,p=0.2)			Dogs.jpg(binarized,p=0.15)		
	precision	recall	F measure	precision	recall	F measure
Ground truth 1	0.6814	0.8246	0.7462	0.4344	0.7616	0.5532
Ground truth 2	0.6395	0.8671	0.7361	0.4875	0.4112	0.4461
Ground truth 3	0.6484	0.8258	0.7264	0.4639	0.5588	0.5069
Ground truth 4	0.6348	0.8714	0.7345	0.3651	0.7634	0.4940
Ground truth 5	0.8597	0.7926	0.8248	0.7733	0.7151	0.7431
Mean	0.69276	0.8363	0.7577	0.5048	0.6420	0.5652

Table1. result for SE detector

	Gallery.jpg			Dogs.jpg		
	precision	recall	F measure	precision	recall	F measure
Ground truth 1	0.2219	0.7215	0.3395	0.0557	0.6537	0.1027
Ground truth 2	0.2029	0.7390	0.3184	0.0977	0.5514	0.1660
Ground truth 3	0.2199	0.7523	0.3403	0.0930	0.7496	0.1655
Ground truth 4	0.2083	0.7679	0.3277	0.0442	0.6186	0.0825
Ground truth 5	0.3026	0.7495	0.4312	0.1157	0.7158	0.1992
Mean	0.2311	0.2878	0.3529	0.08126	0.6578	0.1446

Table2. Result for Sobel edge detector

	Gallery.jpg(binarized,HT=0.4)			Dogs.jpg(binarized,HT=0.3)		
	precision	recall	F measure	precision	recall	F measure
Ground truth 1	0.2059	0.8246	0.3295	0.0844	0.7500	0.1517
Ground truth 2	0.1859	0.8339	0.3039	0.1284	0.5480	0.2081
Ground truth 3	0.2012	0.8480	0.3252	0.1196	0.7292	0.2055
Ground truth 4	0.1819	0.8263	0.2982	0.0670	0.7095	0.1225
Ground	0.2715	0.8282	0.4088	0.1782	0.8344	0.2937

truth 5						
Mean	0.20928	0.2878	0.3345	0.1155	0.71422	0.1988

Table3. Result of Canny edge detection

1.5.2 Discussion

(1)

For structured edge, it's a data driven method. As we can see from the result, it doesn't contains the texture inside the objects. It only shows the main outline. So it's the advantage of structured edge. It can avoid the most of noise detection.Also using sharpen and non-maximum suppression can extract the strongest edge. So SE has the best performance among these three detection. For it's disadvantages, training model before detection might be an inconvenience, usually we need bunch of time to train a model with good performance, especially when the size of forest is large.

For sobel edge detection and canny edge detection, they are more focus on local relationships between pixels. So that they would be interfered by a lot of details(noise) inside objects. Sometimes the texture inside objects might not be the edges we truly want.

To be more specific, sobel edge detection would be effected by noise easily. As we can get from the section1, it can't operate well in the grass part in Dogs.jpg. However, because of the simple core algorithm implementation, it runs really fast and it's easy to operate.

For Canny edge detection, it already implement non-maximum suppression and use two threshold to improve the quality of figure. By using those tech, Canny edge can alleviate part of blur edge and extract strong edge in the image. So that we can see that it's F measure is higher than sobel edge detection. However, for the interference detection, it can't perform well. As we can see from the result, it can't avoid being effected by interference object such as texture on the floor and the grass on the land.

(2) Personally speaking, Gallery.jpg is easier to get a high F measure. First of all, the main edges we need to detect in Gallery.jpg is clear. There exists some interference, for example the picture on the wall, and the texture on the floor and clothes. However, the interference won't affect a lot. When it comes to Dogs.jpg part, the grass on the land and the hair of dogs can affect the result in a great degree. Especially the grass, they can generate huge noise. In sobel edge detection and canny edge detection, the situation can be worse.

(3)

Let's discuss the rationale behind the F measure definition. It is impossible to get a high F measure, if precision is significantly higher than recall, or vice versa.

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

$$assume(P = aR)$$

$$F = \frac{2aR^2}{(1+a)R} = \frac{2}{1+\frac{1}{a}}R = \frac{2}{(1+\frac{1}{a})\frac{1}{R}} = \frac{2}{(\frac{1}{R}+\frac{1}{P})}$$

As we can see from the formula, if either of R or P is really small, the denominator would be really large, so the F would be a small number.

If R+P is a constant number, we have:

$$F = \frac{2 * P * R}{R + P}$$
$$assume(P + R = a)$$
$$F = \frac{2P(a-P)}{a} = \frac{2aP - 2P^2}{a}$$

As we can see that it's a quadratic equation of P, it has maximum when P = 1/2*a, which means P=R.

Problem 2.Digital Half toning

2.1 Abstraction and motivation

Half toning is a tech that can use dots to restore whole continuous tone image. Continuous tone image usually has pixel value in a continuous range, while half tone image only contains 1 and 0. This method relies on optical illusion, which is human's eye can complement 0/1 pattern to continuous pattern.

2.2 Dithering

2.2.1 Approach and Procedure for Fixed thresholding

In fixed thresholding, the threshold is a fixed number. The formula of implementation is given as follow:

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

To be more specific, traverse whole matrix and use a fixed threshold to judge the value of pixel.

2.2.2 Experimental Result

Figure8. result of fixed thresholding

2.2.3 Discussion

Since the threshold is a fixed number, the pixels in a fixed range would be valued as 1 and in other range would be valued as 0. As we can see from the result, the tower can be generated in a good degree. Because the pixels in the tower part can be really different to the pixels around them in a small area, it can cause the dithering affect of eyes. However, when it comes to the cloud part, the fixed thresholding doesn't perform well, because the value of cloud pixels are very similar to each other in a large area.

2.2.4 Approach and Procedure for Random thresholding

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

2.2.5 Experimental Results

Figure9. result of random thresholding

2.2.6 Discussion

In random thresholding, the threshold is a random number. So that the whole result looks like being painted by dots. However, when we look at it at a far distance, the dithering affect would make it looks really close to original picture. It is really good at sightly value changing in a large area. As we can seen from result, the cloud part is much better than fixed thresholding. However, when it comes to tower part, which has sudden changes of pixels value, random thresholding doesn't perform well. Compared with fixed thresholding, it performs much worse, we can't seen any details in the tower.

2.2.7 Approach and Procedure for Dithering Matrix

In dithering matrix, we actually use a matrix to represent different degree of grayscale. Because of dithering effect, we can seen a result as if it contains different grayscale.
For an index matrix whose size is 2 by 2, we have:

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

For lager size matrix, we have induction formula:

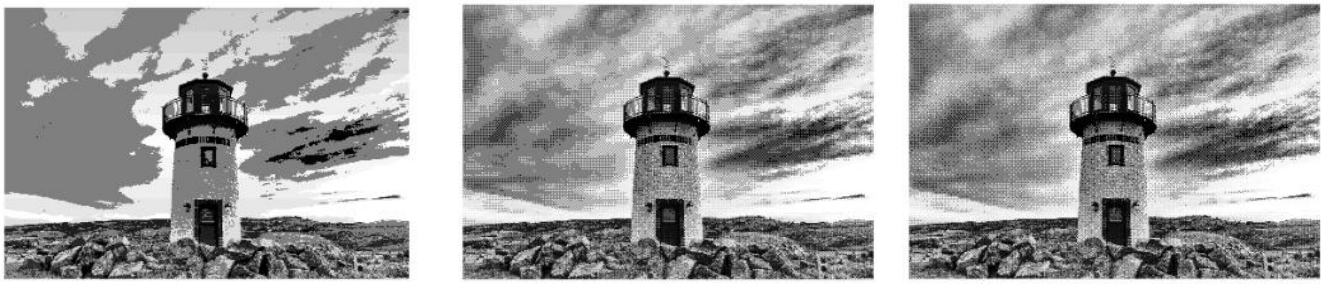
$$I_{2n}(i,j) = \begin{bmatrix} 4 \times I_n(i,j) + 1 & 4 \times I_n(i,j) + 2 \\ 4 \times I_n(i,j) + 3 & 4 \times I_n(i,j) \end{bmatrix}$$

The most important part is the formula to transfer index matrix to threshold matrix:

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

$$G(i,j) = \begin{cases} 0 & if : F(i,j) \le T(i \bmod N, j \bmod N) \\ 255 & otherwise \end{cases}$$

2.2.8 Experimental Result



(a)I2 *(b)I8* *(c) I32*

Figure10. result of dithering matrix

2.2.9 Discussion

As we can seen from the formula in 2.2.7, threshold has positive correlation with index value, which means the low index in the matrix has high probability to get a white pixels. For the induction rule, the larger matrix contains higher numbers in it, which actually represents more different degree of grayscale. For example, in size 2 matrix, there's only 4 different threshold. However, in size 4 matrix, there exists 16 degree grayscale. It's also the reason why using a larger index matrix can get a better performance. Comparing these three figures, which use index matrix(2*2, 8*8, 32*32) respectively, the larger index matrix is, the better result is. In figure...., we can't see any details of tower and cloud. When it comes to 8-by-8 index matrix, we can see the texture of tower and the gradual change in the cloud part. For 32-by-32 index matrix, the details can actually be better. The color changes more naturally in cloud and rock part.

2.3 Error Diffusion

2.3.1 Experimental Result

Figure11. result of error diffusion

In error diffusion, the basic idea is using a error diffusion matrix to pass the error to surrounding pixels. Because of the effect of dithering, surrounding pixels can alleviate the vision error. To get a better result, we can use a larger matrix. It can distribute error to larger range of surrounding pixels, and more accurate. In Floyd-Steinberg's method, the error diffusion matrix is too small, so that the color changes suddenly in result. Compared with the Floyd-Steinberg's method, error diffusion proposed by JJN and Stucki use a larger matrix and the weight in matrix distributed in a better manner. To be more specific, in JJN and Stucki's method, weight in matrix relates with distance between pixels. All the pixels with same distance from the center has the same weight. For Stucki's method, the weight in matrix is really close to the direct proportion to the square of distance. As we can see from

the results, it works. In conclusion, choosing a larger error diffusion matrix and weight relates with distance can get a better performance.

2.4 Color Halftoning with Error Diffusion

2.4.1 Approach and Procedure

For color half-toning with error diffusion part, we have two implementation. The first one is separable error diffusion, in this method, we apply error diffusion on three channels of image separately and combine them at final. Second method is MBVQ-based error diffusion. This method applies error diffusion on three channels at same time, it considered three channel as a whole part. Before applying error diffusion, we need to transfer pixel into minimize brightness variance quadruple first.

2.4.2 Experimental Result for Separable Error Diffusion

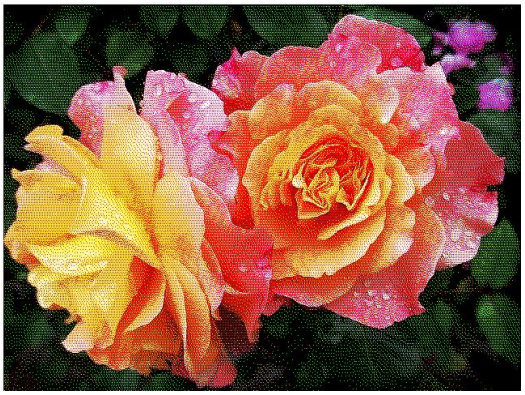


Figure12. result of separable error diffusion

2.4.3 Discussion

In separable error diffusion, we did half-toning operation on CMY channels separately. First of all, the original figure need to be transferred to CMY channel. Since the (Y,B),(C,R),(M,G) are complementary color pairs, the transfer function is simple.

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

After being transferred into CMY channel, we apply Floyd-Steinberg's error diffusion to each channel separately. Finally we use transfer function again to get the result in RGB channels. As we can see, the result is not bad, the boundary between pedals is really clear. We can also see the water drop on petals clearly. However, there still exist some brightness problem on the centre of flower. The boundary of petals seems to be highlighted. The centre part of the right flower and the boundary of petal of left flower are affected in a great degree.

2.4.4 Experimental Results for MBVQ-based Error Diffusion

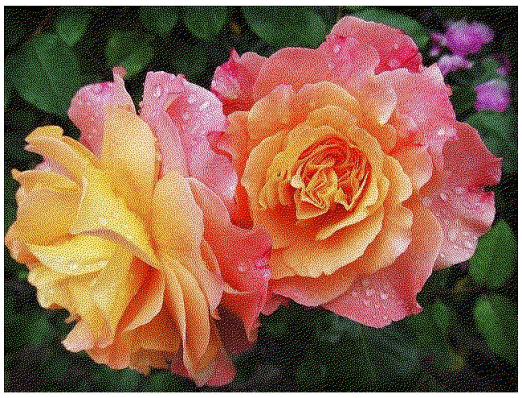
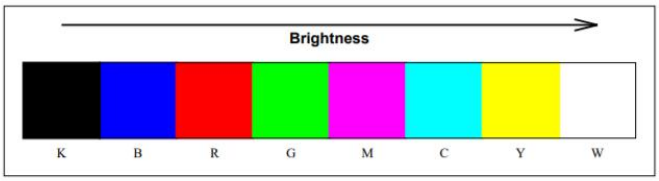


Figure13.result of MBVQ-based Error Diffusion

2.4.5 Discussion

The main problem of separable error diffusion is the brightness problem. The main reason to cause this problem is that, in separable error diffusion, the figure doesn't considered as a whole entity. The half-toning is operated on each channel separately, so that it can't make sure the sudden change of brightness.



As we known from the paper written by D.Shaked, if we only need to consider eight basic color, the order of brightness variance can be shown as above.

The key idea of MBVQ-based error diffusion is using halftone quadruples to minimize the brightness variation. To be more specific, we start to transfer black and white halftone couple to green and magenta. The process ends when there are no white or black halftones. According to the different situation(no white only/no black only/both), we transform black or white and a primary colored halftone to couples of secondary colored halftones. Finally, we get our minimize brightness variance quadruple. This method can minimize the brightness variance while preserving average color. After deciding quadruples for specific pixel, we only use the four colors in quadruples to transform the original color.

As we can see from our implementation, the color of centre part of flower is much more soft than figure12. Also, the color changing is closer to the original picture and more natural.