Problem 1 Edge detection

1. Motivation and abstraction

Edge detection is a basic image processing problem and is fundamental for computer vision. The aim of edge detection is to detect the change of images' discontinuities in depth and surface also the change in texture and change in illuminance. Edge detection can preserve useful information and get rid of the unnecessary one. Edge detection is a crucial technique especially for feature detection.

2. Question and answers

(a) Sobel Edge Detector

a) Method

Sobel edge detector is calculating edges by calculating the x and y gradient to each pixel and then based on these x-gradient and y-gradient, generate the gradient of the original image then set a threshold to determine the edges.

Gradient

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnitude

$$|\nabla f| = \sqrt{g_y^2 + g_x^2}$$

Orientation

$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

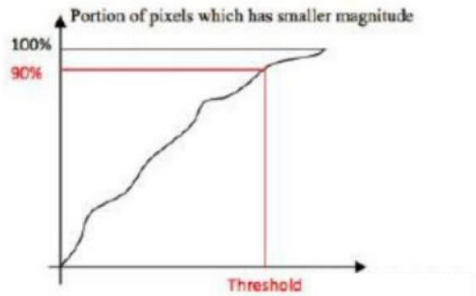
The x-gradient and y-gradient are calculated by the Sobel filters.

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

Gx

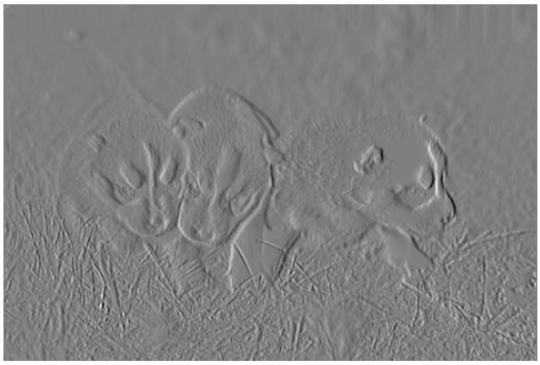
+1	+2	+1
0	0	0
-1	-2	-1

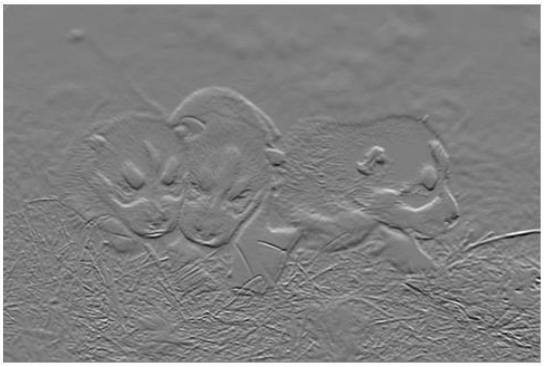
Gy



b) Results and analysis

Original image Dogs.raw

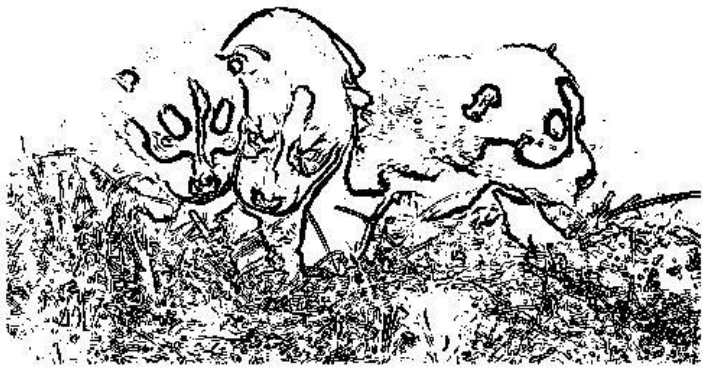




x-gradient **y-gradient**

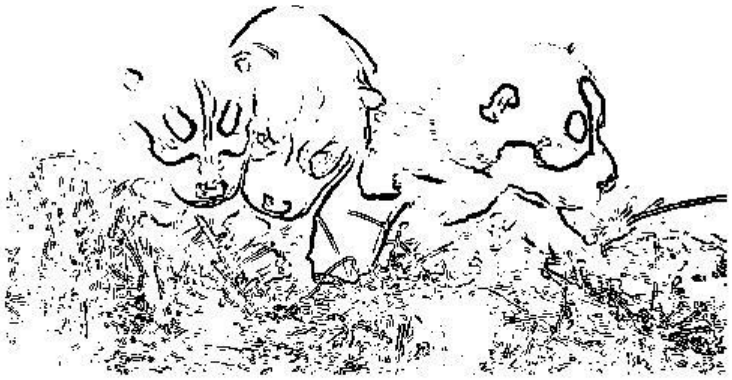
the x y gradient image is normalized to 0-255

Normalized gradient image

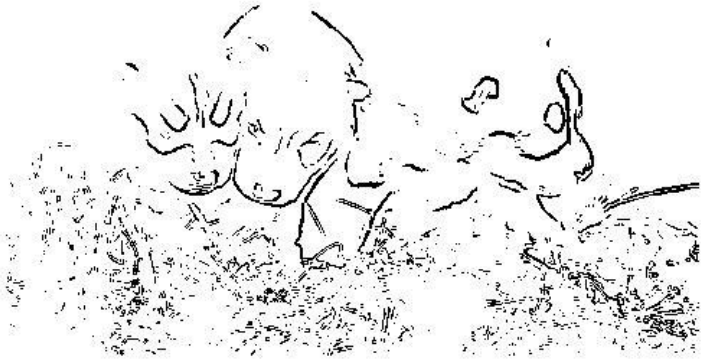


Threshold = 80%

Threshold = 85%

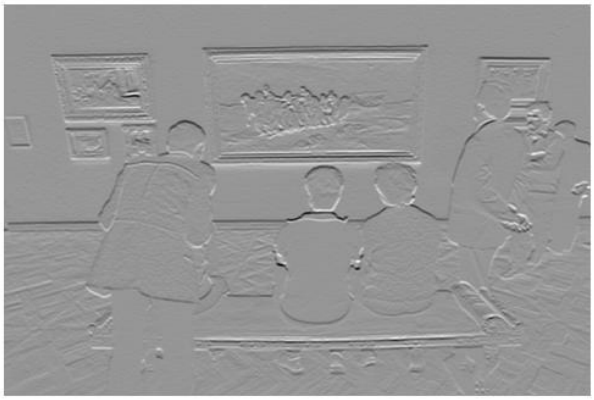


Threshold = 90%



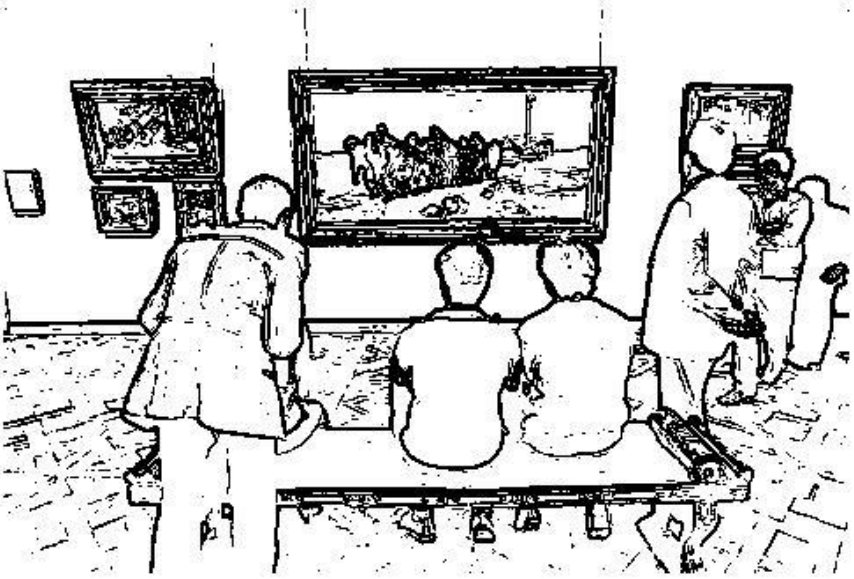
Threshold = 95%

Original image Gallery.raw

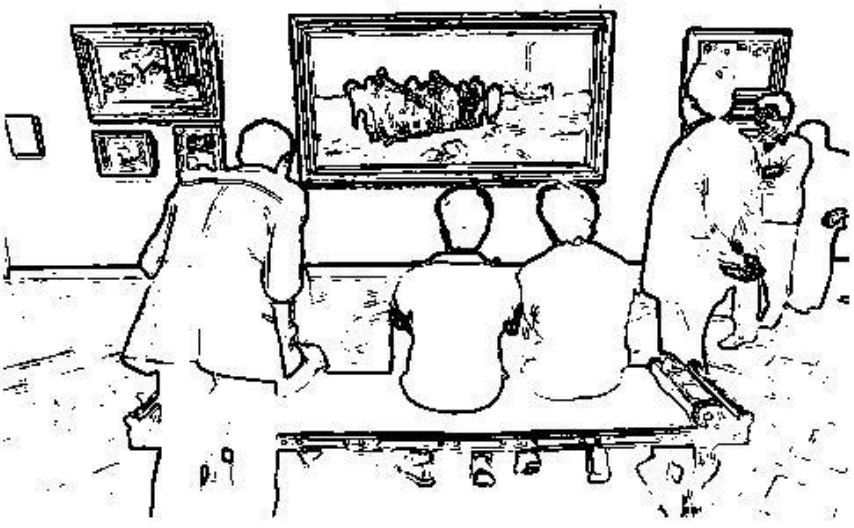


the x y gradient image is normalized to 0-255

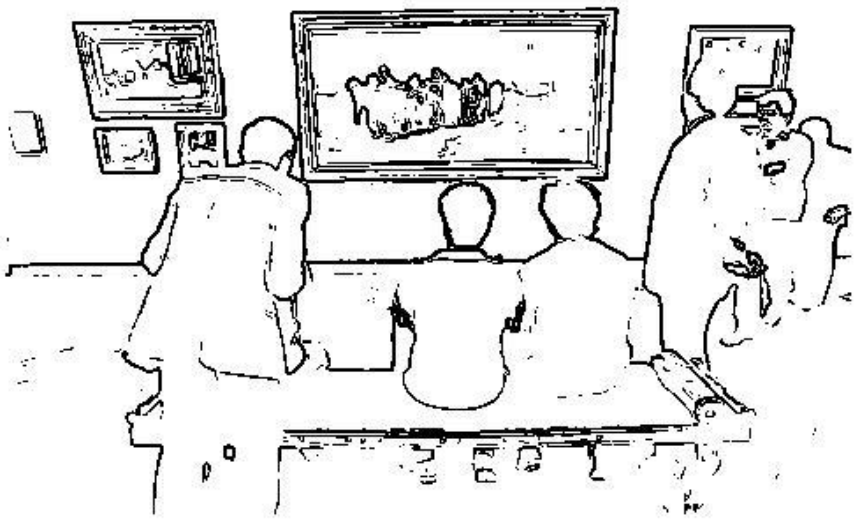
Normalized gradient image



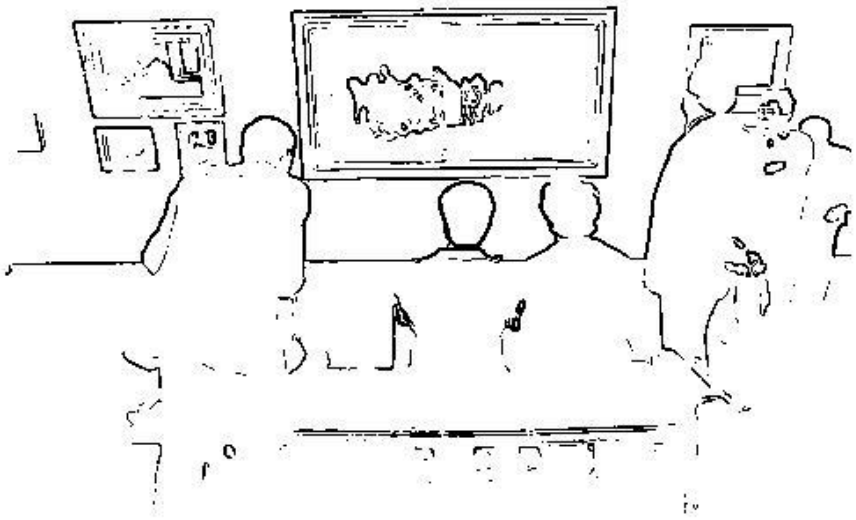
Threshold = 80%



Threshold = 85%



Threshold = 90%



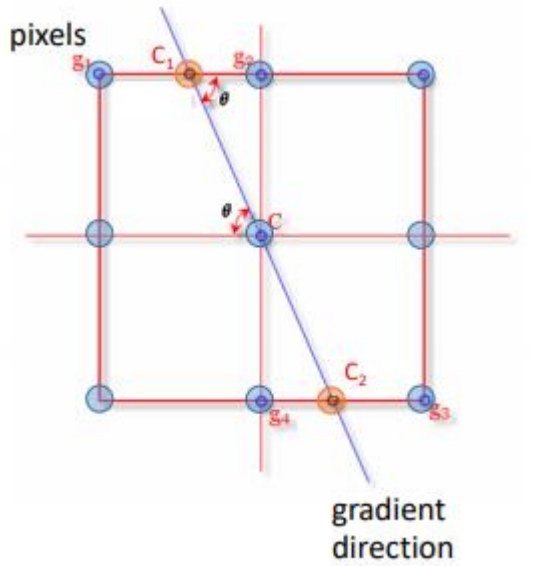
Threshold = 95%

By trying one by one, I choose 92 for Dogs and 90 for Gallery. Since 92 for Dogs could eliminate the grass best and do not lost so much contour of dogs. And 90 for Gallery could show the contour of human well and illuminate unimportant details.

(b) Canny Edge detector

(1) Non-maximum suppression

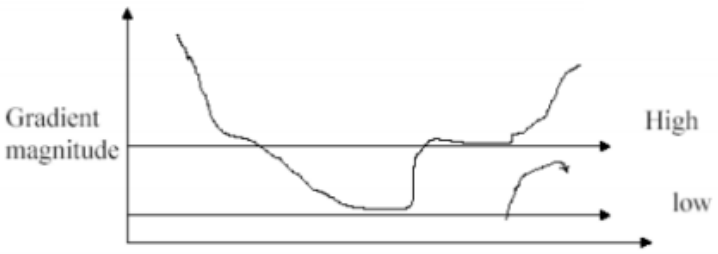
After the calculation of gradient map, non-maximum suppression suppresses all gradient values to 0 expect the local maximum. Also the value of local maximum is preserved to its original value.



$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

Θ is the gradient direction which is calculated by x-gradient and y-gradient, on this direction the edge detection method eliminate all non-maximum pixels. And the value of C1 and C2 are calculated by interpolation of nearby pixels.

(2) Double thresholding



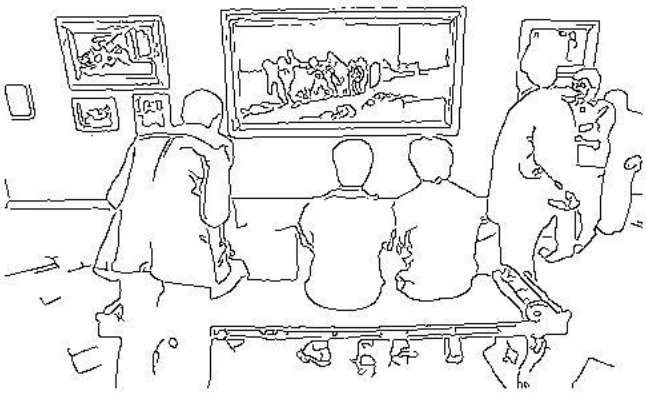
There are two thresholds to be utilized for the edge detection. The way they work is shown below:

i. If a pixel gradient value is greater than the higher value, then this pixel is labeled as an edge.

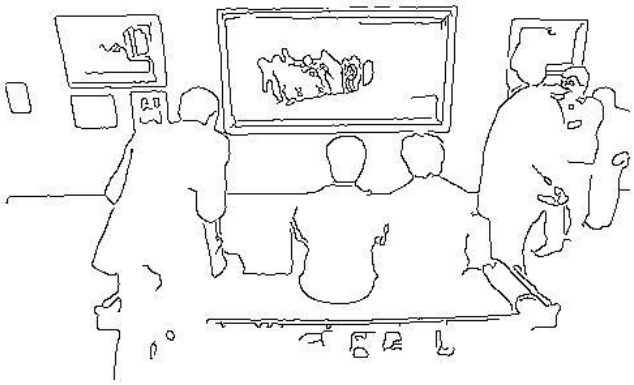
ii. Is a pixel gradient value is smaller than the lower value, then this pixel is not an edge.

iii. If a pixel gradient is between these two thresholds, if the gradient curve has some part greater than the higher thresholding then it is labeled as an edge, otherwise it will be rejected.

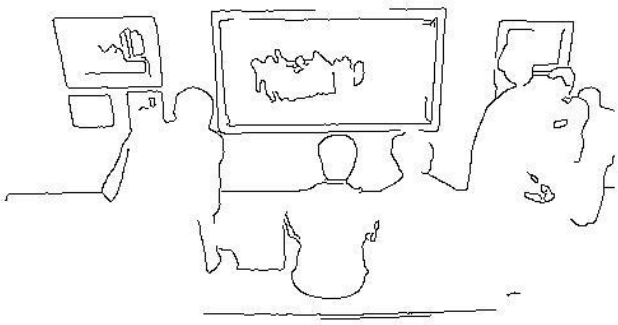
(3) Result and analysis



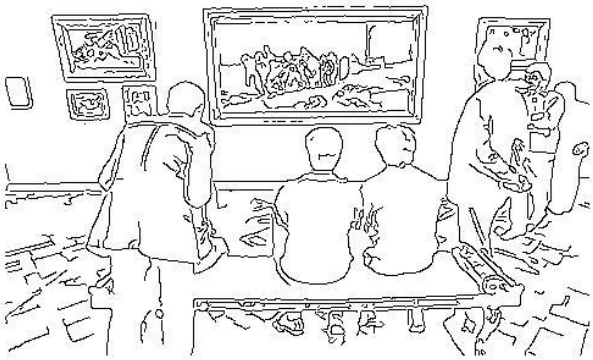
Lower : 0.05 upper 0.15



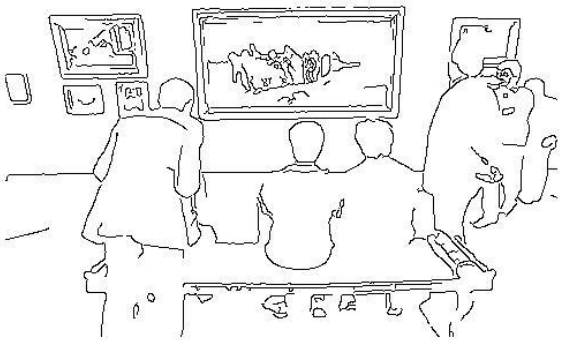
Lower : 0.1 upper 0.3



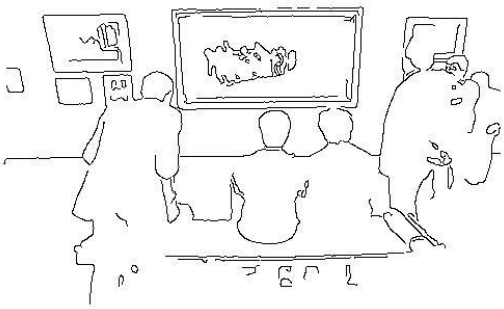
Lower : 0.15 upper 0.45



Lower : 0.05 upper 0.10



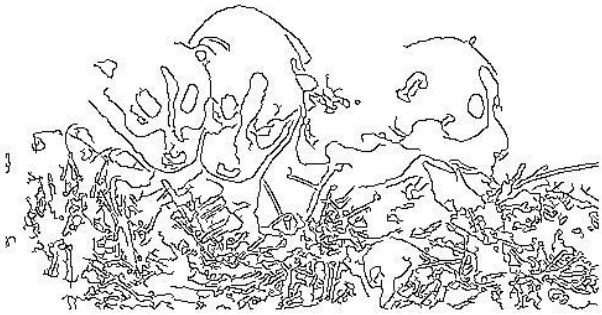
Lower : 0.1 upper 0.20



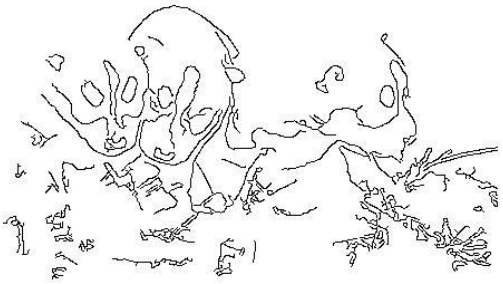
Lower : 0.15 upper 0.30

For Dogs

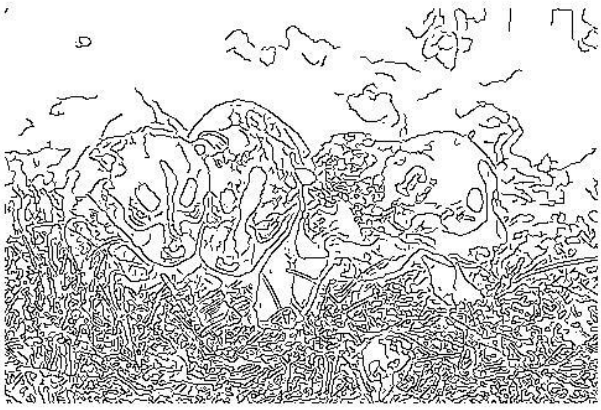
Lower : 0.05 upper 0.15



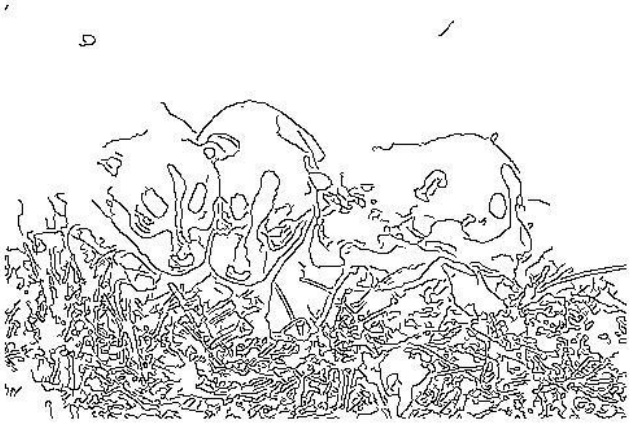
Lower : 0.1 upper 0.3



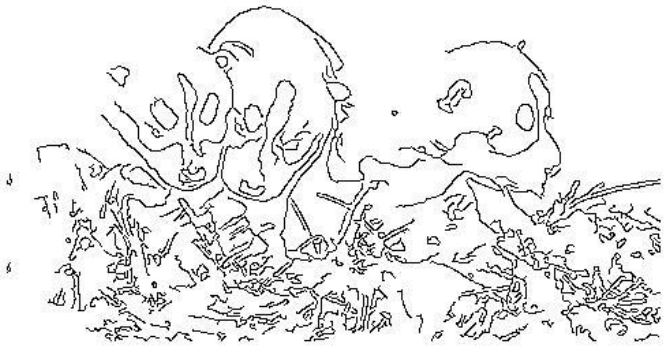
Lower : 0.15 upper 0.45



Lower : 0.05 upper 0.1



Lower : 0.1 upper 0.2



Lower : 0.15 upper 0.3

As the images shown above, for image gallery if the upper threshold is greater than 0.2, then it may miss some of the important boundaries. And if the lower threshold is less than 0.1 it may introduce some unnecessary features for example people in the painting.
For Dogs, if the upper threshold is greater than 0.3, then it may miss some of the important boundaries. And if the lower threshold is less than 0.15 it may introduce some unnecessary features for example in the grass. So these two thresholds are depend on the image to be processed and differ from image to image.

(c) Structure Edge

(1) Structure Edge Algorithm:

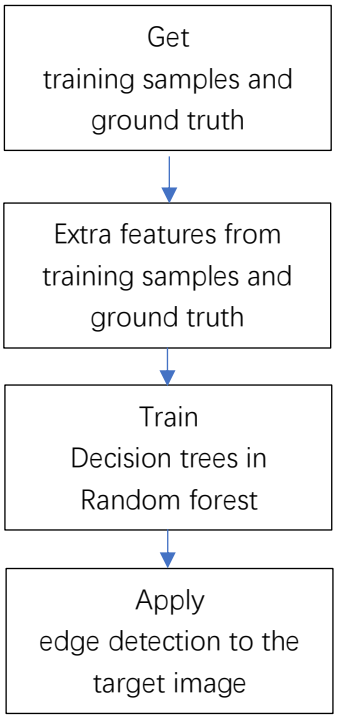
In order to detect edges in the original image, we choose 32*32 image patch to construct a dataset. But there will be 2^(32*32) features in a single path which is too expensive to calculate. We need to do some operations to reduce features in each patch. Here is the algorithm:

Three color channels in CIE-LUV color space are computed into 2 scale with normal magnitude, each of them is split into 4 channels based on orientation. So the total 13 channels are 3 color 2 magnitude and 8 orientation. After this each channel is blurred with radius 2 triangle filter and down sample by 2. Result in 32*32*13/4 = 3328 features. Also apply larger radius blur with radius 8 to each channel and down sampled to 5*5. So, the total features are 3328+300*13=7228 features in each patch.

Also for edge detection there are 16*16 pixels labels for the segmentation mask. There is no need for 2^(16*16) features, since we can down the features by $\binom{256}{2} = 32640$.

Then training decision trees to construct a random forest. These decision trees are trained based on the training set. For each tree, they have a given node j and the training set $S_j \in \mathrm{X} * \Upsilon$ the aim of training is to find the best parameters of θ_j to achieve the split function $h(x, \theta_j)$ which can good classify the data. The ability to classify the data is calculated by the function $I_j = I(S_j, S_j^L, S_j^R) = H(S_j) - \sum \frac{|S_j^K|}{|S_j|} H(S_j^K)$ where the K belong to {L,R}. The decision tree will map the patch input to a target binary edge patch.

The construction of random forest will be introduced in the next part. After the construction of the random forest, this forest can be used to do edge detection for each patch input. And after this, by combination of these output patches we could built the edge image.



(2) Random forest

Random forest is a classifier which is constructed by a mount of sub classifiers which vote to get the classify result. It has some good characters, it could deal with high-dimensional sample and can handle missing date and indicate the most important feature. The random forest gathers decisions made by decision trees and integrate them to make a final decision.

Construction:

If the size of training data is N.

i. training decision tree with n randomly selected n bootstrap samples from the training data set.
ii. If the sample has M different characters, when training the decision node to construct branches in the decision tree, select m characters randomly from the M characters where m<<M. Then use several methods to choose one character of m to construct this node.
iii. The every inter node are all established on step ii until cannot generate more branches. Notice there are no pruning to the decision tree.
iv. Establish amount of decision trees

Principle:

The principle for decision tree is to combine weak classifiers to construct a strong classifier.

(3) Results and parameters selection

Parameters:

Multiscale

This parameter is set to 1 by default. When the value is real the program will utilize structured edge detector for three times and average the results of these three implements to achieve a better result.

Sharpen

This parameter is set to 2 to get a better result. After the classification of random forest some edge may be overlapped due to has less votes. To correct these overlaps the sharpen parameter is used. This parameter can be valued to 0 1 2 The smaller value the faster speed of the program.

nTreesEval

This parameter is set to 4 in my program. This value is the number of

decision trees in the random forest. The larger number the longer time will take to implement the edge detecting program.

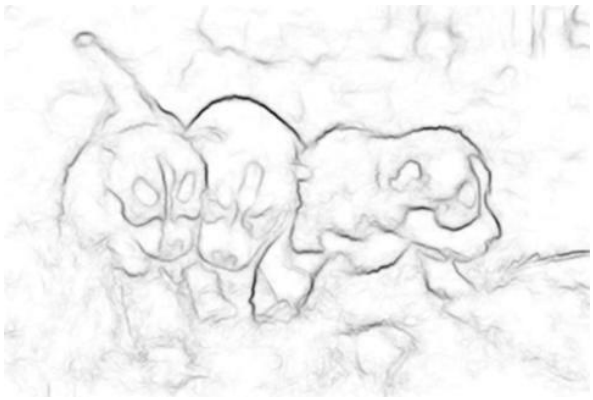
nThreads

This parameter is set by default to 4. The meaning of this parameter is max number threads for evaluation

nms

This parameter indicates to the non-maximun suppression. This algorithm has already been introduced above.

Result:
The parameters chosen is shown by
(multiscale, sharpen,ntreesEval,nThreads,nms)



(0,2,2,4,0)

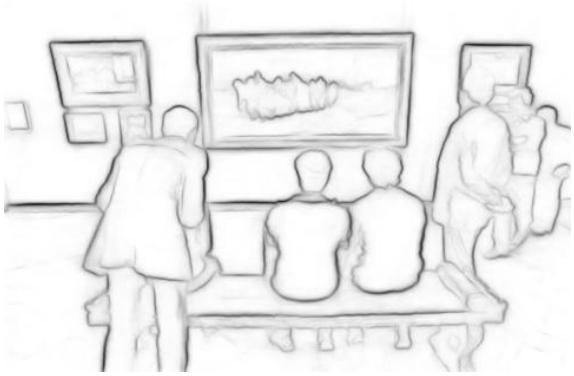
(0,2,4,4,0)

(1,2,2,4,0)

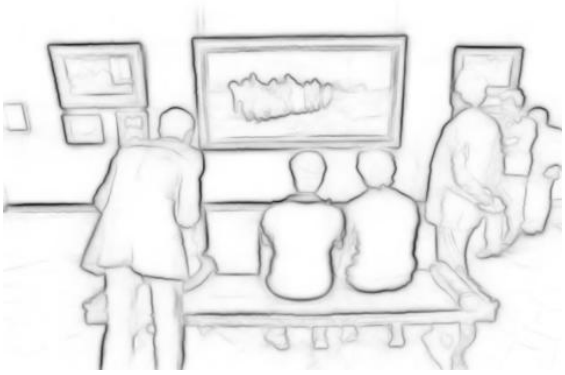
(1,2,4,4,0)

(0,2,2,4,0)

(0,2,4,4,0)



(1,2,2,4,0)



(1,2,4,4,0)

(4) Discussion

When there are more than four trees in the random forest, the result has no visual different comparing to four trees. Structured edge can better eliminate unnecessary information comparing to canny for example grass and texture in both Dogs and Gallery has been weakened. Also structure has a better edge detection on visual since the edge of dogs head and man's leg cannot be detected in canny method.

Also the structured edge method could generate a probability map edge map, but for canny method, it can only generate a binary edge map.

(d) Performance Evaluation

(1) Average score for different ground truth

Average score is calculated by the mean recall and mean precision to each ground truth

Canny:

L = lower threshold U = upper threshold

DOGs	GT1	GT2	GT3	GT4	GT5
L=0.05 U=0.10	0.1858	0.3258	0.2280	0.1564	0.3219
L=0.10 U=0.20	0.2663	0.3418	0.3041	0.2293	0.4667
L=0.15 U=0.30	0.3579	0.3476	0.3567	0.3024	0.6178
L=0.20 U=0.40	0.4535	0.3433	0.3900	0.3862	0.7341

Gallery	GT1	GT2	GT3	GT4	GT5
L=0.05 U=0.10	0.5853	0.5487	0.5632	0.5353	0.7034
L=0.10 U=0.20	0.6698	0.6482	0.6446	0.6418	0.7894
L=0.15 U=0.30	0.6425	0.6457	0.6259	0.6714	0.7718
L=0.20 U=0.40	0.6134	0.6380	0.5880	0.7057	0.6944

Sobel： P=percentage of pixels

Dogs	GT1	GT2	GT3	GT4	GT5
P=80%	0.1328	0.1784	0.2001	0.1087	0.2620
P=85%	0.1542	0.2040	0.2325	0.1268	0.2983
P=90%	0.1854	0.2253	0.2535	0.1556	0.3337
P=95%	0.2309	0.2363	0.2742	0.1951	0.3850

Gallery	GT1	GT2	GT3	GT4	GT5
P=80%	0.4727	0.4356	0.4500	0.4286	0.5690
P=85%	0.5433	0.5068	0.5184	0.4974	0.6506
P=90%	0.5744	0.5538	0.5560	0.5528	0.7037
P=95%	0.5764	0.5743	0.5515	0.5940	0.6901

Structure edge

(multiscale, sharpen,ntreesEval,nThreads,nms)

Dogs	GT1	GT2	GT3	GT4	GT5
(1,2,2,4,0)	0.2067	0.1056	0.1472	0.2280	0.1234
(0,2,4,4,0)	0.2993	0.1592	0.2181	0.3095	0.2119
(0,2,2,4,1)	0.2997	0.1596	0.2186	0.3243	0.2092
(1,2,4,4,0)	0.2016	0.1031	0.1435	0.2274	0.1258

Gallery	GT1	GT2	GT3	GT4	GT5
(1,2,2,4,0)	0.4758	0.5173	0.4559	0.4722	0.4129
(0,2,4,4,0)	0.5476	0.5645	0.5354	0.5543	0.5243
(0,2,2,4,1)	0.5364	0.5567	0.5222	0.5679	0.5332
(1,2,4,4,0)	0.4828	0.5260	0.4636	0.4797	0.4187

For Dogs, the fifth ground truth has the highest F value for every method, and for Gallery the second ground truth has the highest F value for every method. So when evaluating the performance of different algorithm, the chosen of grand truth takes an important role in it.

secondly the mean F over ground truth is given below

thresholding			0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Dogs	canny	L0.05 U0.1	0.243601	0.243575	0.243599	0.243625	0.243574	0.243576	0.243574	0.243555	0.243554
		L0.1 U0.2	0.321677	0.321603	0.321598	0.321677	0.321471	0.321597	0.321593	0.321636	0.321636
		L0.15 U0.3	0.396516	0.396288	0.396514	0.396499	0.396514	0.396462	0.396514	0.396514	0.396514
		L0.2 U0.4	0.4615	0.461092	0.461569	0.461239	0.461569	0.461362	0.461409	0.461569	0.461569
	sobel	80	0.17639	0.17639	0.17637	0.17639	0.176368	0.17639	0.17639	0.17639	0.176349
		85	0.203152	0.203152	0.203176	0.203102	0.20315	0.20315	0.203152	0.203176	0.203152
		90	0.230736	0.230736	0.230736	0.230736	0.230736	0.230736	0.230704	0.230704	0.230736
		95	0.264321	0.264273	0.264225	0.264273	0.264273	0.264321	0.264273	0.264321	0.264321
	structure	(0,2,2,4,0)	0.161963	0.162193	0.162193	0.162193	0.162193	0.162193	0.162193	0.162193	0.162193
		(0,2,4,4,0)	0.23961	0.23961	0.23961	0.23961	0.23961	0.23961	0.239456	0.239497	0.23961
		(1,2,2,4,0)	0.242317	0.242317	0.242317	0.242317	0.242317	0.242317	0.242317	0.242074	0.242317
		(1,2,2,4,0)	0.160181	0.160344	0.160115	0.160344	0.160344	0.160344	0.160181	0.160344	0.160344
Gallery	canny	L0.05 U0.1	0.587178	0.587116	0.587181	0.587235	0.587181	0.587263	0.587207	0.587178	0.587203
		L0.1 U0.2	0.678956	0.678666	0.678738	0.678853	0.678772	0.678816	0.678811	0.678812	0.678883
		L0.15 U0.3	0.671328	0.671624	0.671373	0.671493	0.671353	0.67135	0.671402	0.671586	0.671403
		L0.2 U0.4	0.648092	0.64773	0.648008	0.648123	0.648412	0.647862	0.647958	0.647855	0.648089
	sobel	80	0.471172	0.471213	0.471235	0.471191	0.471168	0.471172	0.471149	0.471194	0.471214
		85	0.543255	0.543306	0.543333	0.543307	0.543333	0.543333	0.543335	0.543259	0.543359
		90	0.588089	0.588187	0.588094	0.587962	0.588065	0.588131	0.588154	0.588159	0.588094
		95	0.597263	0.597349	0.597196	0.597192	0.597305	0.597313	0.597183	0.597262	0.597276
	structure	(0,2,2,4,0)	0.467024	0.466762	0.466768	0.466951	0.466879	0.466698	0.466952	0.466677	0.466698
		(0,2,4,4,0)	0.545035	0.545375	0.545149	0.545253	0.545063	0.545375	0.545269	0.545181	0.545113
		(1,2,2,4,0)	0.543425	0.54339	0.543139	0.543289	0.543294	0.543227	0.543175	0.543294	0.543149
		(1,2,2,4,0)	0.474004	0.473701	0.474192	0.473927	0.473986	0.473984	0.474108	0.473911	0.47406

box with pink has the highest F value.

For dogs, the thresholding with highest F score for canny is 0.4 and for Sobel is 0.4 and 0.5 for structure edge.
For Gallery, the thresholding with highest F score for canny is 0.2 and for Sobel is 0.2 and 0.2 for structure edge.

The performance of each edge detection methods vary greatly because of the parameters chosen. But based on the F value shown above Canny behave better than structured edge than Sobel method.

(2) Gallery could get higher F measure. Because in the Dogs the boundary of dogs is hard to be detected needless to say dogs in grass. Grass is like a camouflage clothing which has a complex texture and will broke the contour of dog. Also the grass will influence the detection of edges, since its luminance changes dramatically. But for gallery, every texture is simple and pure for example the wall, so the boundary is significant and can be easily characterize. This is the reason why gallery may be easier to achieve a higher F score.

(3)

If the precision is significantly high which means the false positive is nearly 0. But if the recall is too low, the F value will equal to the recall value which may decrease F value significantly. Vice versa. If the sum of precision and recall is a constant then, F is linearly depending on the product of P and R. As we know,

$$\mathrm{P} + \mathrm{R} \geq \sqrt{P * R}$$

The equal sign is established if and only if P=R. So when wanting to achieve the highest value of F, P and R must equal. Proofed.

$$\text{Precision}: P = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}}$$

$$\text{Recall}: R = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}}$$

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

Problem 2: Digital Half-toning

1. Motivation

Digital half-toning is a process which transfer gray scale images to binary images or color image into three different binary image channels. The reason for digital halftoning is that, printers only have inks of four different colors which are not able to print out 8-bites gray scale image and 24-bites color images. The printer simulates different levels of brightness by changing the density and size of the print dots. This processing is called digital half-toning.

2. Question and answers

Original image

a) Dithering

(1) Fix thresholding

a. Method

Choose a fix threshold T, if the pixels' gray level of the original image is bigger than T, it will be mapped to 255 otherwise 0. At here we choose T=128.

b. Result

(2) Random thresholding

a. Method

To break the monotones result from the fixed thresholding, we introduce the random thresholding. Each pixel has its own threshold T.

The T is a random number from 0 to 255. The threshold is independent from pixel to pixel. For each pixel, if it is smaller than T map it to 0, otherwise 255.

The random threshold is generated by ***rand(1)*255*** command in Matlab.

b. Result

(3) Dithering thresholding

a. Method

Dithering thresholding has a mask matrix which is generated from a basic index matrix. $I_2 = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$ where 0 means the pixel is most likely to be mapped to 255. And for 3 which is most like to be mapped to 0. And the Bayer index matrix is calculated by the following 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 threshold matrix will be transferred from the index matrix by the following formula.

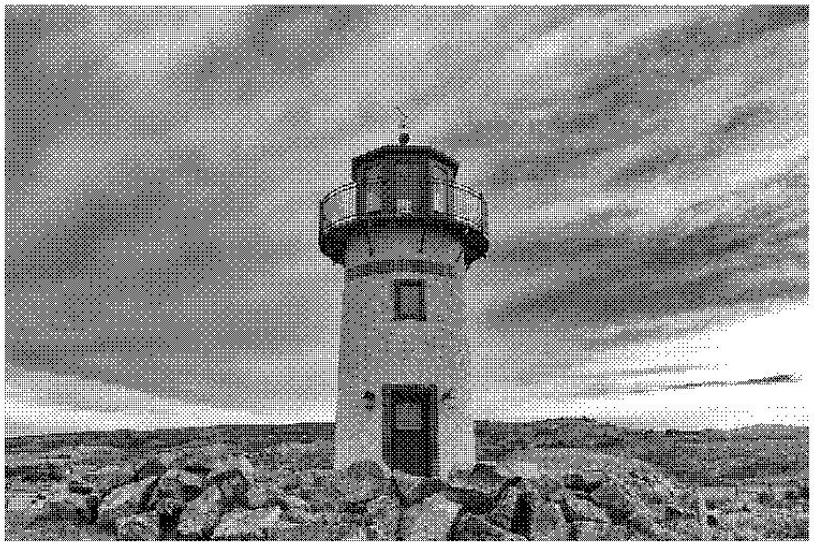
$$T(x,y) = \frac{I_N(x,y) + 0.5}{N^2} \times 255$$

N is the total number of pixels in the threshold matrix. Comparing the threshold matrix periodically in the original matrix in full image. If the gray

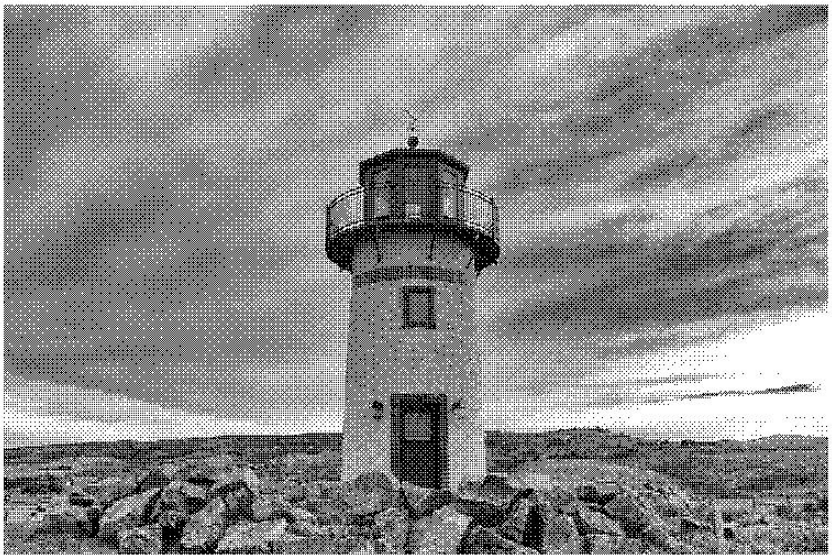
scale from the original image is lower than the threshold then it will be turned to 0, otherwise 255.
In this question we will test I_2, I_8, I_{32}

b. Results

$\boldsymbol{I_2}$



$\boldsymbol{I_8}$



$\boldsymbol{I_{32}}$

(4) Compare the results

Due to the fixed threshold, the fixing thresholding has a monotones result, so it is far from the original image.

For the Random thresholding, because the thresholding is random, so it will introduce noises to the result image. Though it may seem better than the fixed thresholding method, it is still not good for loss detail of the original image.

For the dithering matrix, it is much better when comparing to the two method above. Different index matrix may generate different results. I_2 has the minimum matrix order. Which may cause the losses of image detail. I_{32}has the maximum matrix order, which lead to the bigger stripe in image. This kind of bigger stripe may interrupt the impression of the result. So comparing to $I_2\ and\ I_{32}$, I_8preserve image detail much better and has smaller strip.

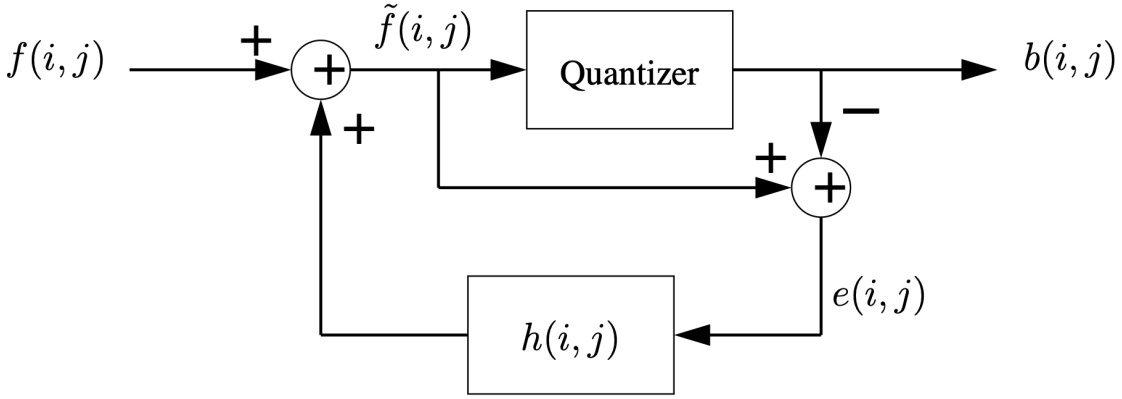
In all among this five output images, I_8has the best output.

b) Error Diffusion

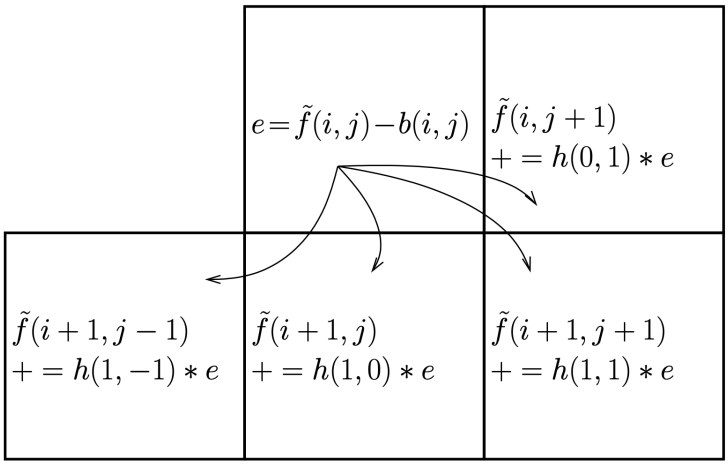
a. Method

The method is shown below.

Where the threshold of quantizer is 128/255(0.5/1) in this method.

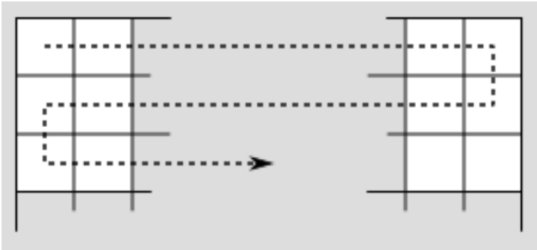


The quantization error $e(i,j) = f'(i,j) - b(i,j)$will be diffused to the nearby pixels according to the following box.



To diffuse error as comprehensive as possible, in this problem we will use serpentine s**can**ning to diffuse the error.

- **Serpentine parsing**

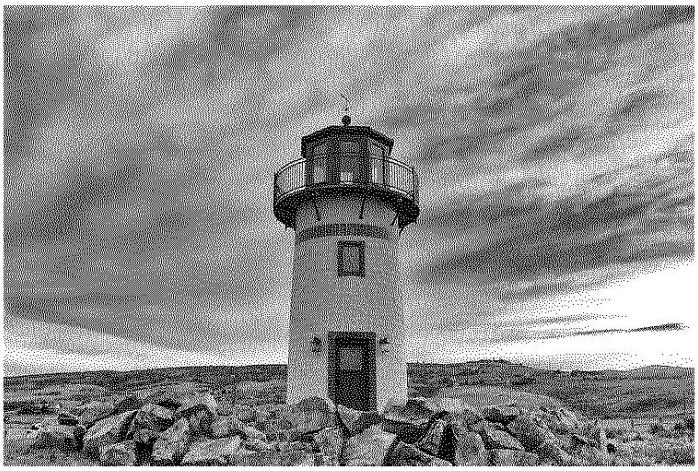


b. Result

Here we will test three different kind of error diffusions.

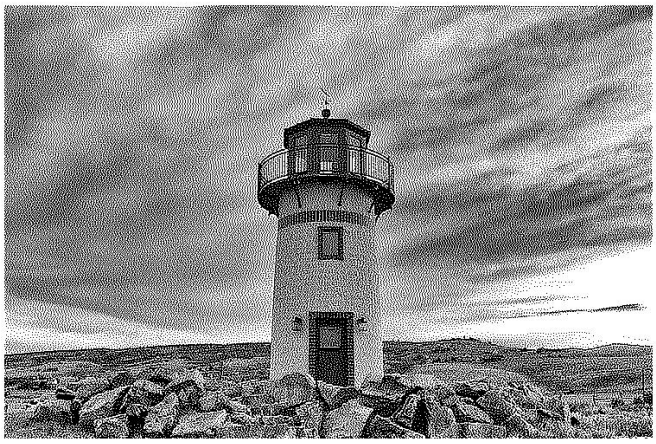
(1) Floyd-Steinberg's error diffusion

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



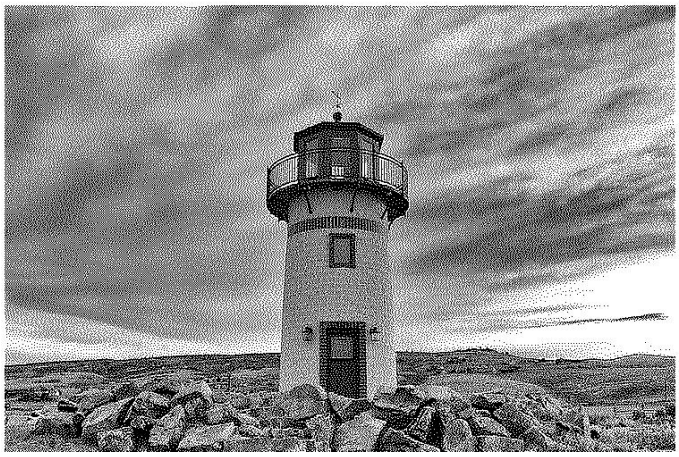
(2) Error diffusion proposed by Jarvis, Judice, and Ninke (JJN)

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



(3) Error diffusion proposed by Stucki

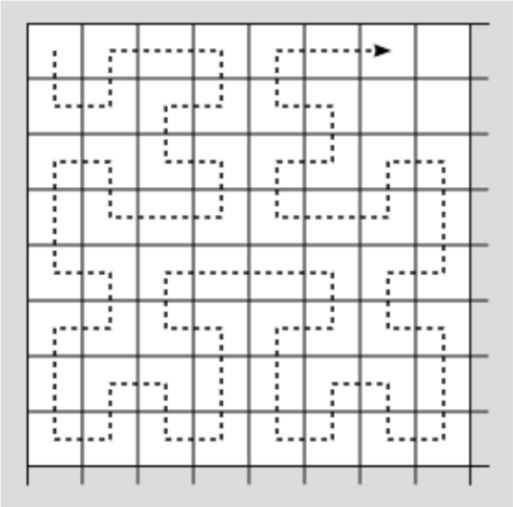
$$\frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$



c. Questions answering

Comparing to the dithering matrix method, I prefer this error diffusion method. Since it preserves the detail of the image well and look like the original image. To get a better result, we may utilize Hilbert curve to diffuse the errors, which may more equally diffuse the errors in the whole image.

- **Hilbert curve**



Comparing to serpentine curve which may sediment errors to the pixels with higher row number or moving direction, Hilbert will travel the whole image periodically, and diffuse errors to pixels nearby include the pixel above. This is what the serpentine curve cannot realize. So Hilbert curve is better.

c) Color Halftoning

Original image

a. Motivation

Color image halftoning is different from mono-color halftoning, since it has three different color scales. And they are cyan, yellow and magenta.

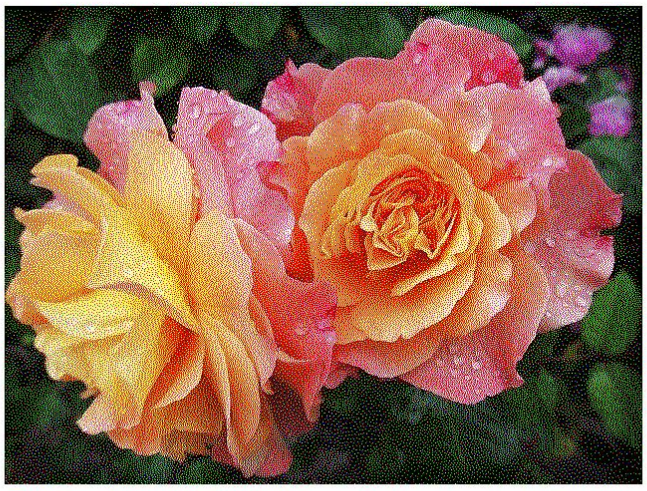
The transfer between and RGB is shown below. Since in electronic form, we cannot show the image printed on paper with cyan magenta and yellow ink. At here we use RGB colors instead of CMY to present the image.

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

b. Methods and results

(1) Separable Error Diffusion

This method does color halftoning separately to each color channel.



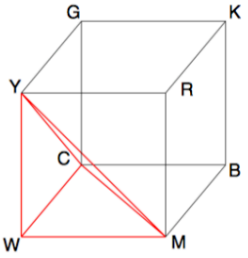
Output image

Shortcoming: This method ignores the luminance information of the original image, which lead to the change of color and change of luminance(energy).

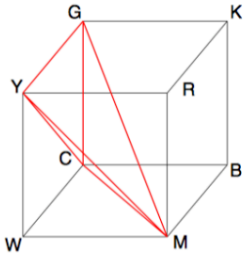
(2) MBVQ-based Error diffusion

1) Method description.

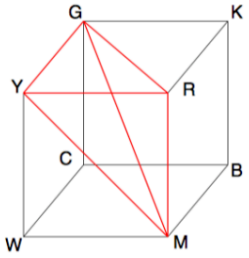
This method quantizes three color channels based on minimum brightness variation quadrants. Based on different RGB values, there are six different quadrants.



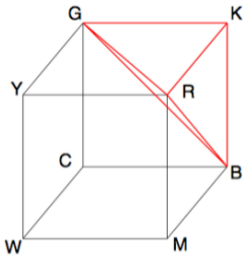
CMYW



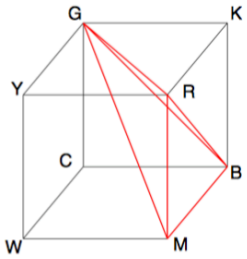
MYGC



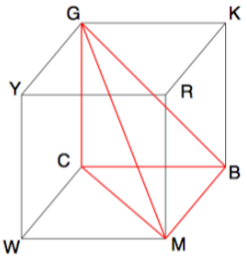
RGMY



KRGB



RGBM



CMGB

Each pixel is determined by the following:

```
pyramid MBVQ(BYTE R, BYTE G, BYTE B)
{
  if((R+G) > 255)
    if((G+B) > 255)
      if((R+G+B) > 510)    return CMYW;
      else                 return MYGC;
    else                   return RGMY;
  else
    if(!((G+B) > 255))
      if(!((R+G+B) > 255)) return KRGB;
      else                 return RGBM;
    else                   return CMGB;
}
```

And then in each quadrant, the final RGB value is determined by its closet vertex.

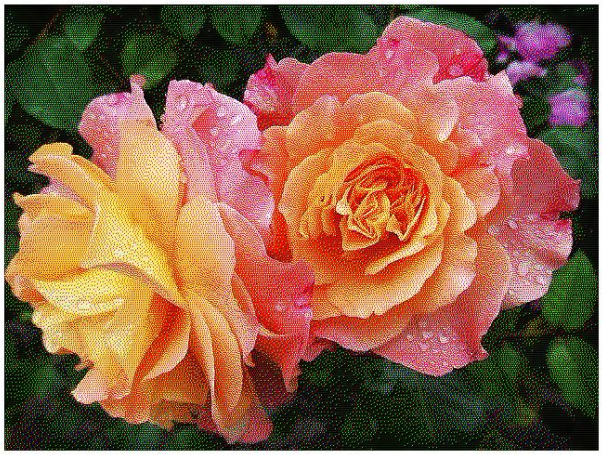
W=(1,1,1) Y=(1,1,0) M=(1,0,1) C=(0,1,1)

R=(1,0,0) G = (0,1,0) B = (0,0,1) K=(0,0,0)

And then diffuse the quantize error separately to RGB channels to the nearby pixels.

This method preserves the energy of original pixels well so it will lead to a better result.

2) Implement and result



Compared with separable error diffusion method, the output image of this method looks closer to the real image. Brightness and color saturation are well preserved, so this method is better