EE 569 HOMEWORK 3

Issued: 10/9/2015 Due: 11/1/2015

Name: Meiyi Yang

Email: meiyiyan@usc.edu

USC ID: 6761-0405-85

Problem 1. Geometrical Modification

1.1 Motivation

This problem go through the problem of geometrical modification of digital image in 2-Dimension and 3-Dimension. In problem A, a swirling effect is implemented to *Kate* image. In problem b, 3D cube is mapped from world coordination to camera coordination. The basic operation of geometrical modification using there are translation, rotation and scaling.

1.2 Approach and procedure

1.2.1 Swirl Effect

Translation, rotation, and scaling. Here I use 4D transformation including translation, rotation and scaling to implement the effect of image and adjust the image while mapping. The matrix of **T S R** is showing below [1]:

$$T(tx,ty,tz]\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & tx\\0 & 1 & 0 & ty\\0 & 0 & 1 & tz\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$Rx(\theta)\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\0 & cos\theta & -sin\theta & 0\\0 & sin\theta & cos\theta & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$Ry(\theta)\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}cos\theta & 0 & sin\theta & 0\\0 & 1 & 0 & 0\\-sin\theta & 0 & cos\theta & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$Rz(\theta)\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}cos\theta & -sin\theta & 0 & 0\\sin\theta & cos\theta & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$S(sx, sy, sz)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The workflow of implement a geometrical effect is shown in Figure 1.

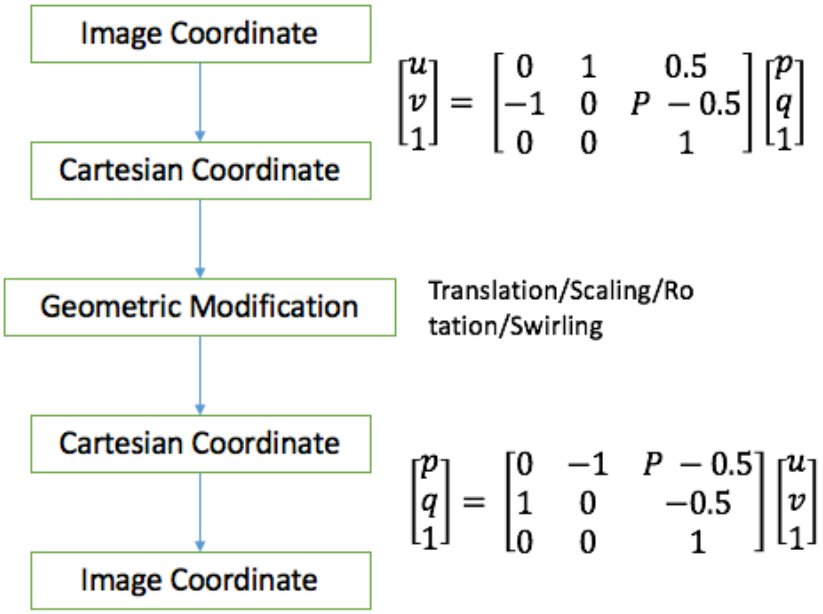


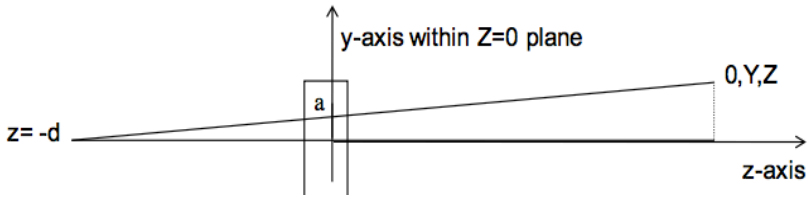
Figure 1. Workflow of geometrical effect

The implement of swirling is rotation the image, the degree is determined by the distance from the pixel to the center of the image. One thing need to be consider is that the image must translate half of the image distance to make sure the image is in the center of the Cartesian Coordinate.

1.2.2 Perspective transformation and image geometry

Perspective projection and image geometry. In Figure 2, the perspective projection of (0, Y, Z) occurs at a (0, y, 0) (Z = 0 plane). The camera focal point is at (0, 0, -d) [1]. Then we can get

So the projection transformation are:



$$x = \frac{X}{\frac{Z}{d} + 1}$$

$$y = \frac{Y}{\frac{Z}{d} + 1}$$

$$z = \frac{Z}{\frac{Z}{d} + 1}$$

$$[X, Y, Z, (Z/d) + 1] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Transform the world coordinate to camera coordinate. The transformation includes two steps: from world coordinate to camera coordinate, from camera coordinates to image plan coordinate.

Apply two matrixes to the [X, Y, Z, 1]^T [2]:

$$Extrinsic\ camera\ matrix: [R|t] = \begin{bmatrix} XcX & XcY & XcZ & -\boldsymbol{r} \cdot \boldsymbol{Xc} \\ YcX & YcY & YcZ & -\boldsymbol{r} \cdot \boldsymbol{Yc} \\ ZcX & ZcY & ZcZ & -\boldsymbol{r} \cdot \boldsymbol{Zc} \end{bmatrix}$$

$$Intrinsic\ camera\ matrix: K = \begin{bmatrix} f & 0 & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{bmatrix}$$

$$Image\ Plane: w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K\,[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The parameters using here are:

$$\boldsymbol{Xc} = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)^T$$

$$\boldsymbol{Yc} = (\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, -\frac{2}{\sqrt{6}})^T$$

$$\boldsymbol{Zc} = (-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})^T$$

$$\boldsymbol{r} = (5, 5, 5), \qquad f = \sqrt{3}$$

$$cx = imagewidth / 2, \qquad imagewidth = 200\,pixels * 0.02;$$
$$cy = imageheight / 2, \qquad imageheight = 200\,pixels * 0.02;$$

Pre-processing the image. The implementation of getting the image data from the cube images is first get the data of one image, mapping the image to the world coordinate by some transformations (details in Figure 2). Then combine six image dataset to one dataset. For example, the top image should be in the (-1, -1, 1), (-1, 1, 1), (1, -1, 1), (1, 1, 1) four vertexes.

```
Image img_ori0(200, 200, 3, "p1_image/baby.raw");
img_ori0.Initial_Geodata();
img_ori0.Effect_Translation(-100.0f, -100.0f, 1.0f);
img_ori0.Effect_Rotation(90.0f, MODE_Z);
img_ori0.Effect_Scaling(0.01, 0.01, 1.0f);

Image img_ori1(200, 200, 3, "p1_image/baby_bear.raw");
img_ori1.Initial_Geodata();
img_ori1.Effect_Translation(-100, -100, 1);
img_ori1.Effect_Rotation(90, MODE_Y);
img_ori1.Effect_Rotation(90, MODE_X);
img_ori1.Effect_Scaling(1.0f, 0.01, 0.01);

Image img_ori2(200, 200, 3, "p1_image/baby_cat.raw");
img_ori2.Initial_Geodata();
img_ori2.Effect_Translation(-100, -100, 1);
img_ori2.Effect_Rotation(-90, MODE_X);
img_ori2.Effect_Rotation(180, MODE_Y);
img_ori2.Effect_Scaling(0.01, 1.0f, 0.01);
```

Figure 2. Examples of mapping the input image.

Reverse mapping. To reverse mapping, the image plane coordinate to the world coordinate. All the matrixes are used instead by the inverse matrixes. First apply inverse intrinsic matrix, then inverse extrinsic matrix. One of the thing must be careful is Z buffer. When reverse mapping, sometimes one image may hide another because they have different depth. The most common method here to use is Z-buffer. Just use a Z-buffer to remove the hidden surfaces for each pixel. Compare computed Z pixel values again current Z value. If the Z pixel value is smaller than Z buffer, write the new pixels to the new image.

1.3 Results

1.3.1 Swirling Effect

The results of swirling are shown in Figure 3. The same parameter to *Lena* image is turn 90 degrees in X axis. Here is also use -90, -45, 45, 90 to compare different parameter for swirling effect.

Also Figure 4 shows the effect of rotation in 30, 60, 90 degree, which will be discuss later in 1.4.1. Figure 5 shows the effect of scaling in (1.5, 1, 1), (0, 1.5, 1). This will be discussing in 1.4.1. Figure 6 is the result of translation. This is an effect using frequently in problem 2b.

Figure 3. Swirling Kate. Swirling degree (-90, -45, 45, 90)

Figure 4. Rotation Kate. Rotation degree (30, 60)

Figure 5. Scaling Kate. Scaling parameter. Top-left: 0.5, 1.0, 1.0. Top-right: 1.0, 0.5, 1.0. Bottom-left: 1.0, 1.5, 1.0. Bottom-right: 1.5, 1.0, 1.0

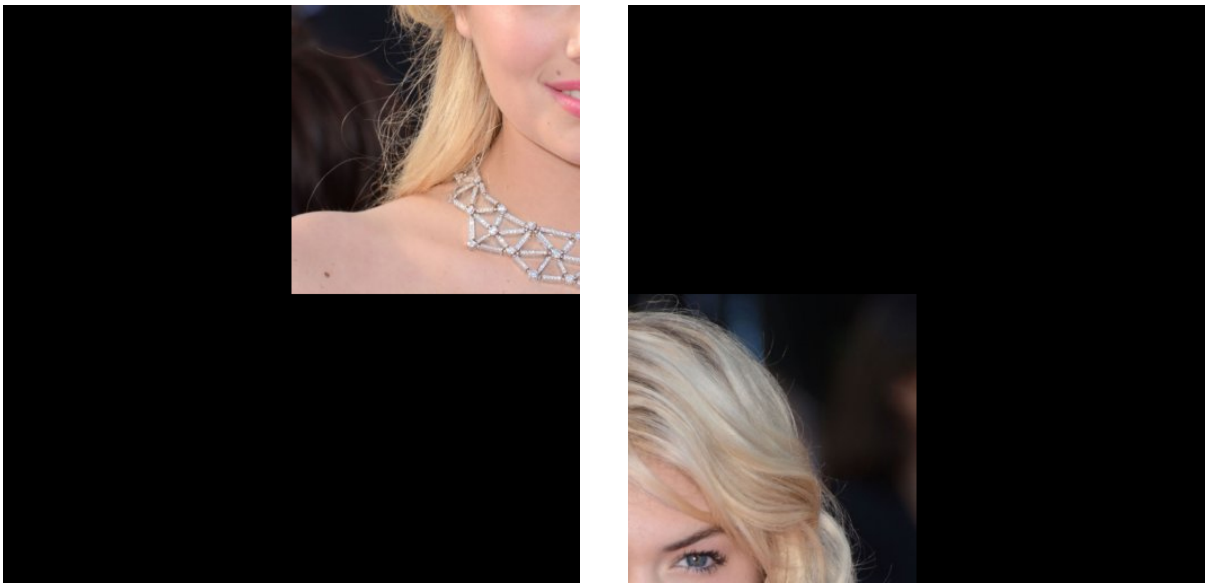


Figure 6. Translation Kate. Parameter. Left: -256, -256. Right: 256, 256.

1.3.2. Perspective transformation and imaging geometry

Part of the outputs the world coordinate are shown in Figure 7.

```
Image image0: 200, 200, 3
(-0.995 -0.995 1 1) (-0.995 -0.985 1 1) (-0.995 -0.975 1 1) (-0.995 -0.965 1 1)
(-0.985 -0.995 1 1) (-0.985 -0.985 1 1) (-0.985 -0.975 1 1) (-0.985 -0.965 1 1)
(-0.975 -0.995 1 1) (-0.975 -0.985 1 1) (-0.975 -0.975 1 1) (-0.975 -0.965 1 1)
(-0.965 -0.995 1 1) (-0.965 -0.985 1 1) (-0.965 -0.975 1 1) (-0.965 -0.965 1 1)
(-0.955 -0.995 1 1) (-0.955 -0.985 1 1) (-0.955 -0.975 1 1) (-0.955 -0.965 1 1)
(-0.945 -0.995 1 1) (-0.945 -0.985 1 1) (-0.945 -0.975 1 1) (-0.945 -0.965 1 1)
(-0.935 -0.995 1 1) (-0.935 -0.985 1 1) (-0.935 -0.975 1 1) (-0.935 -0.965 1 1)
(-0.925 -0.995 1 1) (-0.925 -0.985 1 1) (-0.925 -0.975 1 1) (-0.925 -0.965 1 1)
(-0.915 -0.995 1 1) (-0.915 -0.985 1 1) (-0.915 -0.975 1 1) (-0.915 -0.965 1 1)
(-0.905 -0.995 1 1) (-0.905 -0.985 1 1) (-0.905 -0.975 1 1) (-0.905 -0.965 1 1)
(-0.895 -0.995 1 1) (-0.895 -0.985 1 1) (-0.895 -0.975 1 1) (-0.895 -0.965 1 1)
(-0.885 -0.995 1 1) (-0.885 -0.985 1 1) (-0.885 -0.975 1 1) (-0.885 -0.965 1 1)
(-0.875 -0.995 1 1) (-0.875 -0.985 1 1) (-0.875 -0.975 1 1) (-0.875 -0.965 1 1)
(-0.865 -0.995 1 1) (-0.865 -0.985 1 1) (-0.865 -0.975 1 1) (-0.865 -0.965 1 1)
(-0.855 -0.995 1 1) (-0.855 -0.985 1 1) (-0.855 -0.975 1 1) (-0.855 -0.965 1 1)
(-0.845 -0.995 1 1) (-0.845 -0.985 1 1) (-0.845 -0.975 1 1) (-0.845 -0.965 1 1)
(-0.835 -0.995 1 1) (-0.835 -0.985 1 1) (-0.835 -0.975 1 1) (-0.835 -0.965 1 1)
(-0.825 -0.995 1 1) (-0.825 -0.985 1 1) (-0.825 -0.975 1 1) (-0.825 -0.965 1 1)
(-0.815 -0.995 1 1) (-0.815 -0.985 1 1) (-0.815 -0.975 1 1) (-0.815 -0.965 1 1)
(-0.805 -0.995 1 1) (-0.805 -0.985 1 1) (-0.805 -0.975 1 1) (-0.805 -0.965 1 1)
(-0.795 -0.995 1 1) (-0.795 -0.985 1 1) (-0.795 -0.975 1 1) (-0.795 -0.965 1 1)
(-0.785 -0.995 1 1) (-0.785 -0.985 1 1) (-0.785 -0.975 1 1) (-0.785 -0.965 1 1)
(-0.775 -0.995 1 1) (-0.775 -0.985 1 1) (-0.775 -0.975 1 1) (-0.775 -0.965 1 1)
(-0.765 -0.995 1 1) (-0.765 -0.985 1 1) (-0.765 -0.975 1 1) (-0.765 -0.965 1 1)
(-0.755 -0.995 1 1) (-0.755 -0.985 1 1) (-0.755 -0.975 1 1) (-0.755 -0.965 1 1)
(-0.745 -0.995 1 1) (-0.745 -0.985 1 1) (-0.745 -0.975 1 1) (-0.745 -0.965 1 1)
(-0.735 -0.995 1 1) (-0.735 -0.985 1 1) (-0.735 -0.975 1 1) (-0.735 -0.965 1 1)
(-0.725 -0.995 1 1) (-0.725 -0.985 1 1) (-0.725 -0.975 1 1) (-0.725 -0.965 1 1)
(-0.715 -0.995 1 1) (-0.715 -0.985 1 1) (-0.715 -0.975 1 1) (-0.715 -0.965 1 1)
(-0.705 -0.995 1 1) (-0.705 -0.985 1 1) (-0.705 -0.975 1 1) (-0.705 -0.965 1 1)
```

Figure 7. Output of World coordinate of first image.

Image	Top-Left	Top-Right	Bottom-Left	Bottom-Right
1	(-1, -1, 1)	(-1, 1, 1)	(1, -1, 1)	(1, 1, 1)
2	(1, -1, 1)	(1, 1, 1)	(1, -1, -1)	(1, 1, -1)
3	(1, 1, 1)	(-1, 1, 1)	(1, 1, -1)	(-1, 1, -1)
4	(1, -1, 1)	(-1, -1, 1)	(1, -1, -1)	(-1, -1, -1)
5	(-1, 1, 1)	(-1, -1, 1)	(-1, 1, -1)	(-1, -1, -1)

Table 1. World Coordinate

Results of forward mapping in Figure 8.

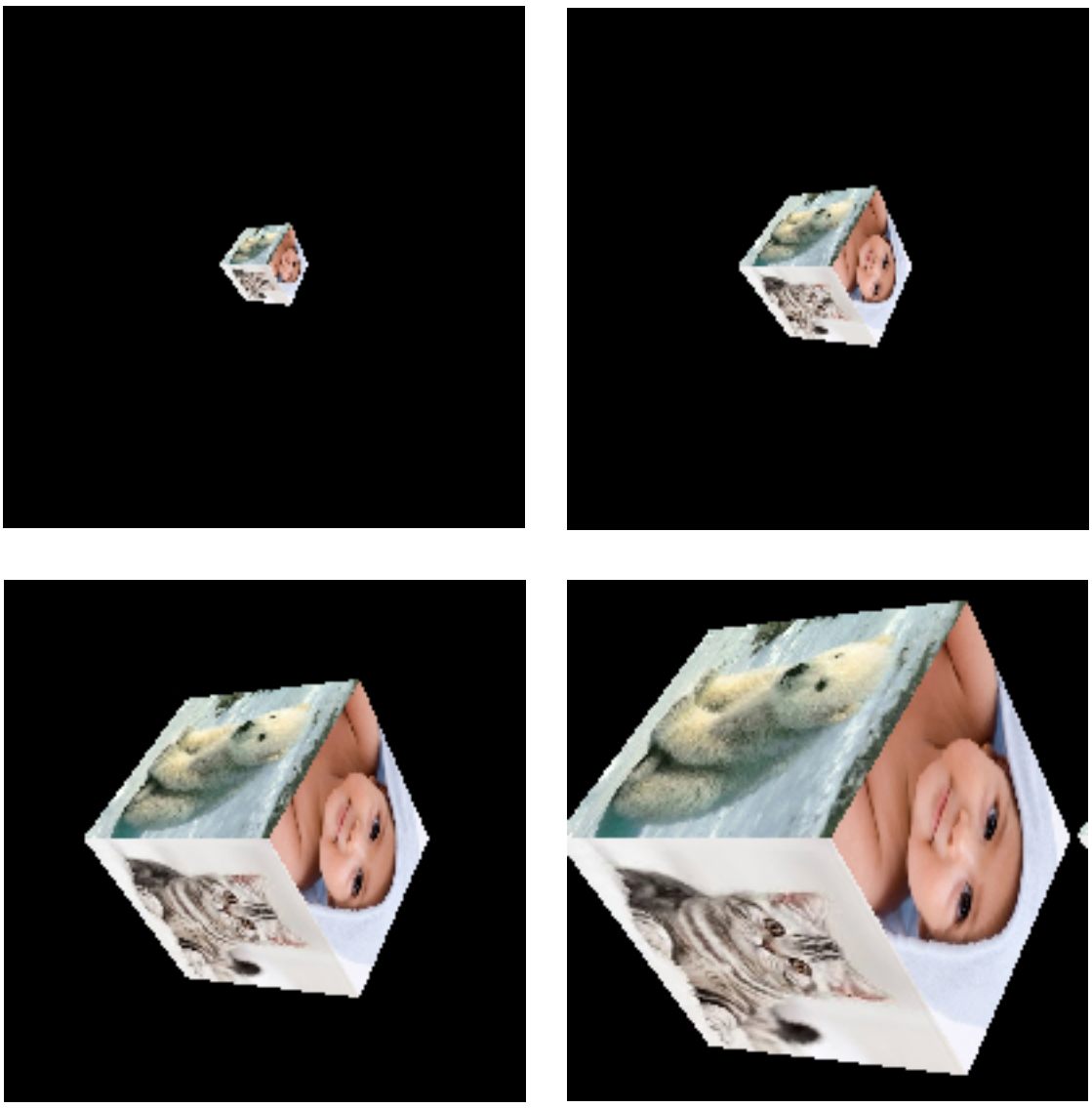


Figure 8. 3D cube. Density = (50, 100, 200, 300)

Figure 9, 10 shows some results which adjust the parameter of f and r.

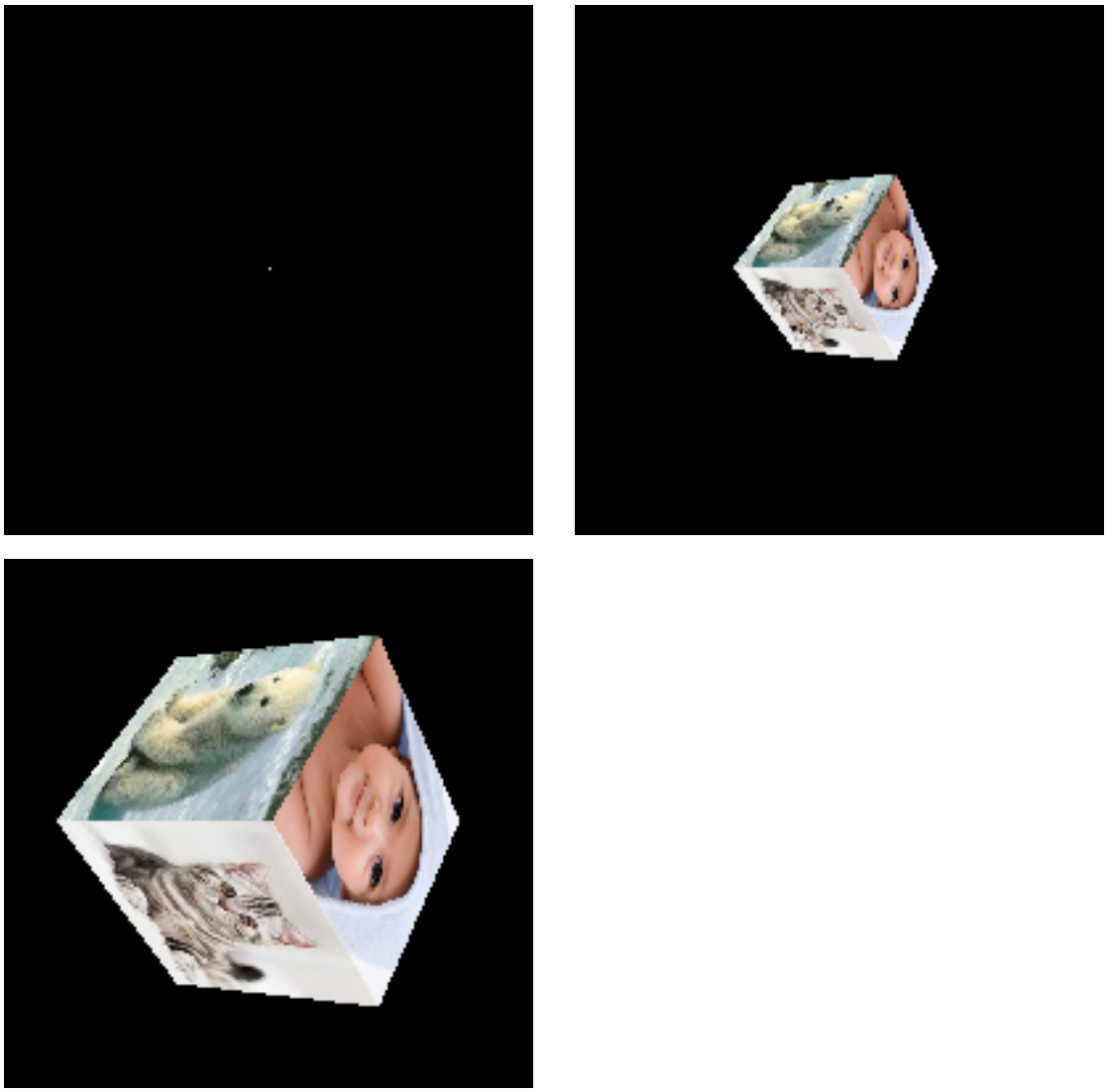


Figure 9. 3D cube. Density = 200, f = (0, 1, 2)

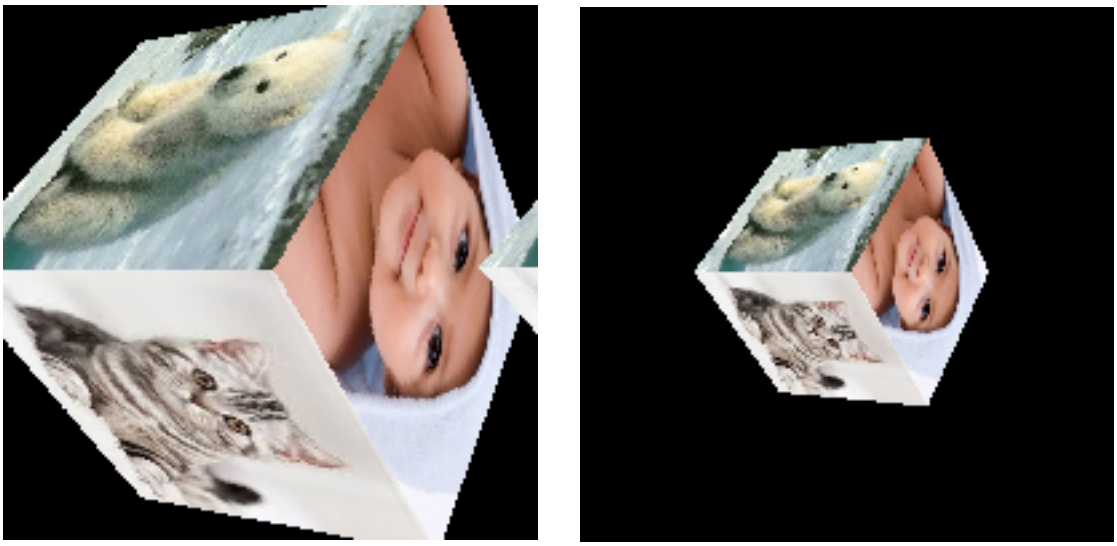


Figure 10. 3 D cube. Density, r = (3, 3, 3), (6, 6, 6)

Reverse Mapping. Figure 11, 12 shows the result of reverse mapping. Figure 11 is the image result using Z-buffer. Figure 12 shows the data result.

Figure 11. Reverse 3D cube

```
Image Extrinsic: 600, 200
(-0.995 -0.995 1 1) (-0.995 -0.985 1 1) (-0.995 -0.975 1 1) (-0.994999 -0.965 1 1)
(-0.985 -0.995 1 1) (-0.985 -0.985 1 1) (-0.985 -0.975 1 1) (-0.985 -0.964999 1 1)
(-0.975 -0.995 1 1) (-0.975 -0.985 1 1) (-0.974999 -0.974999 1 1) (-0.974999 -0.965 1 1)
(-0.965 -0.994999 1 1) (-0.965 -0.985 1 1) (-0.965 -0.974999 1 1) (-0.964999 -0.964999 1 1)
(-0.954999 -0.994999 1 1) (-0.955 -0.985 1 1) (-0.954999 -0.974999 1 1) (-0.955 -0.965 1 1)
(-0.945 -0.995 1 1) (-0.945 -0.984999 1 1) (-0.945 -0.974999 1 1) (-0.945 -0.964999 1 1)
(-0.934999 -0.994999 1 1) (-0.934999 -0.985 1 1) (-0.934999 -0.974999 1 1) (-0.935 -0.965 1 1)
(-0.925 -0.995 1 1) (-0.925 -0.984999 1 1) (-0.925 -0.975 1 1) (-0.924999 -0.965 1 1)
(-0.914999 -0.994999 1 1) (-0.915 -0.985 1 1) (-0.915 -0.974999 1 1) (-0.915 -0.965 1 1)
(-0.905 -0.995 1 1) (-0.904999 -0.985 1 1) (-0.904999 -0.975 1 1) (-0.905 -0.964999 1 1)
(-0.895 -0.994999 1 1) (-0.895 -0.985 1 1) (-0.895 -0.974999 1 1) (-0.894999 -0.964999 1 1)
(-0.885 -0.995 1 1) (-0.884999 -0.985 1 1) (-0.884999 -0.974999 1 1) (-0.885 -0.965 1 1)
(-0.874999 -0.995 1 1) (-0.875 -0.984999 1 1) (-0.875 -0.974999 1 1) (-0.875 -0.964999 1 1)
(-0.864999 -0.994999 1 1) (-0.865 -0.985 1 1) (-0.864999 -0.974999 1 1) (-0.865 -0.965 1 1)
(-0.855 -0.994999 1 1) (-0.855 -0.984999 1 1) (-0.855 -0.975 1 1) (-0.855 -0.964999 1 1)
(-0.844999 -0.994999 1 1) (-0.845 -0.985 1 1) (-0.844999 -0.974999 1 1) (-0.844999 -0.965 1 1)
(-0.835 -0.995 1 1) (-0.834999 -0.985 1 1) (-0.835 -0.974999 1 1) (-0.835 -0.964999 1 1)
(-0.824999 -0.994999 1 1) (-0.824999 -0.985 1 1) (-0.824999 -0.974999 1 1) (-0.824999 -0.965 1 1)
(-0.815 -0.994999 1 1) (-0.815 -0.984999 1 1) (-0.815 -0.974999 1 1) (-0.815 -0.965 1 1)
(-0.804999 -0.994999 1 1) (-0.804999 -0.985 1 1) (-0.805 -0.974999 1 1) (-0.805 -0.965 1 1)
(-0.794999 -0.995 1 1) (-0.795 -0.985 1 1) (-0.795 -0.975 1 1) (-0.795 -0.965 1 1)
(-0.784999 -0.994999 1 1) (-0.785 -0.985001 1 1) (-0.785 -0.975 1 1) (-0.784999 -0.964999 1 1)
(-0.775 -0.995 1 1) (-0.775 -0.985 1 1) (-0.775 -0.974999 1 1) (-0.775 -0.965 1 1)
(-0.764999 -0.995 1 1) (-0.765 -0.984999 1 1) (-0.765 -0.974999 1 1) (-0.764999 -0.964999 1 1)
(-0.755 -0.994999 1 1) (-0.755 -0.985 1 1) (-0.755 -0.974999 1 1) (-0.755 -0.965 1 1)
(-0.744999 -0.995 1 1) (-0.745 -0.984999 1 1) (-0.744999 -0.975 1 1) (-0.745 -0.965 1 1)
(-0.735 -0.994999 1 1) (-0.735 -0.985 1 1) (-0.735 -0.974999 1 1) (-0.735 -0.965 1 1)
(-0.724999 -0.995 1 1) (-0.724999 -0.985 1 1) (-0.724999 -0.975 1 1) (-0.725 -0.964999 1 1)
(-0.715 -0.995 1 1) (-0.715 -0.985 1 1) (-0.715 -0.975 1 1) (-0.714999 -0.965 1 1)
(-0.705 -0.995 1 1) (-0.704999 -0.985 1 1) (-0.705 -0.975 1 1) (-0.704999 -0.965 1 1)
(-0.695 -0.994999 1 1) (-0.695 -0.985 1 1) (-0.695 -0.974999 1 1) (-0.695 -0.964999 1 1)
(-0.684999 -0.995 1 1) (-0.684999 -0.985 1 1) (-0.684999 -0.974999 1 1) (-0.684999 -0.965 1 1)
(-0.675 -0.994999 1 1) (-0.674999 -0.984999 1 1) (-0.675 -0.974999 1 1) (-0.674999 -0.964999 1 1)
(-0.664999 -0.994999 1 1) (-0.665 -0.985 1 1) (-0.664999 -0.974999 1 1) (-0.664999 -0.965 1 1)
(-0.655 -0.995 1 1) (-0.654999 -0.985 1 1) (-0.655 -0.974999 1 1) (-0.654999 -0.964999 1 1)
(-0.645 -0.994999 1 1) (-0.645 -0.985 1 1) (-0.644999 -0.974999 1 1) (-0.645 -0.965 1 1)
(-0.635 -0.995 1 1) (-0.635 -0.984999 1 1) (-0.634999 -0.975 1 1) (-0.634999 -0.965 1 1)
(-0.625 -0.994999 1 1) (-0.625 -0.985 1 1) (-0.625 -0.974999 1 1) (-0.625 -0.965 1 1)
(-0.614999 -0.995 1 1) (-0.614999 -0.985 1 1) (-0.615 -0.975 1 1) (-0.615 -0.964999 1 1)
(-0.605 -0.994999 1 1) (-0.605 -0.985 1 1) (-0.604999 -0.974999 1 1) (-0.605 -0.965 1 1)
(-0.595 -0.994999 1 1) (-0.594999 -0.984999 1 1) (-0.594999 -0.974999 1 1) (-0.595 -0.965 1 1)
(-0.584999 -0.994999 1 1) (-0.585 -0.985 1 1) (-0.585 -0.975 1 1) (-0.585 -0.965 1 1)
```

Figure 12. Reverse Data of up image

1.4 Discussion

1.4.1 Swirling effect

There are two things need to be consider in swirling effects.

First is using bilinear interpolation to show the fractional data.

Second is if part of image is out of the range of display, set that part of image to 0. One of the special case is the whole image is out of the range of display, we must check this case first. The out-of-range control can be see in Figure 4.

Figure 5 show the scaling effect, we can see the bilinear interpolation preserve the image detail.

1.4.2 Perspective transformation

Density. In figure 8, we can find smaller density get smaller result. And the image of smaller density is not as clear as the higher density image.

Focal length f. To analysis the effect of focal length, I set the f to 0. Then we can the result become one point. As the focal length increasing, the image become bigger.

Camera vector r. In figure 10, r has the similar influence to the image as f. I use an "wrong" image in Figure 10 r = (3, 3, 3). We can see is the 3D cube is out of the range of display, it will have a apparently artifact.

Improvement. Because the 3D cube image is not as clear as the original image. I think there must be some filter can be used to improve the clearness of the image. For example, bilinear interpolation, which has been used in this case.

Reverse mapping. As discussed in the 1.2.2, reverse mapping must use Z-buffer. Or the result will be a merge image of up image and bottom image.

Problem 2. Digital Halftoning

2.1. Motivation

Halftoning is widely used in the reprographic technique. It simulates continuous grayscale of color image to dots.[3]. In this problem, I implement four method of halftoning. Two for grayscale images: Dithering, and Error Diffusion. Two for RGB images: Scalar Color Halftoning, and Vector Color Halftoning.

2.2 Approach and procedure

2.2.1 Dithering

Dithering. The image we want to generate is a binary image. Intuitionally, we only need to binary the original image by a threshold such as T = 127 to binary the image. But the result is not as good as we think. We can try to add some noise to the image to get a more clear result, but because halftoning is a technique widely used in printing, we need a effective and quick algorithm. Adding Gaussian noise is slow and waste of time. Here we use dithering to add some noise to the image.

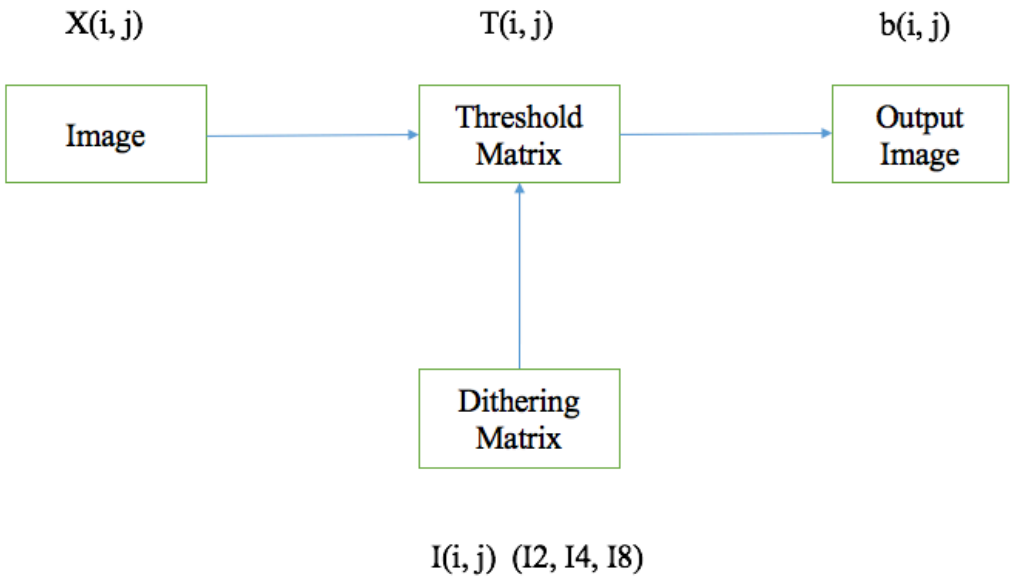


Figure 13. Workflow of dithering

The implementation of dithering is shown in Figure 13. The common matrixes used in I matrix are I_2, I_4, I_8. The three index matrixes and threshold matrix T can be seen in Figure 14. T matrix is

$$T(i,j) = 255 * \frac{I(i,j) + 0.5}{N^2}$$

$$b(i,j) = \begin{matrix} 255 \; if \; X(i,j) > T \\ 0 \; otherwise \end{matrix}$$

```
Bayer_Matrix: size 2
1 2
3 0

Bayer_Matrix: size 4
5 9 6 10
13 1 14 2
7 11 4 8
15 3 12 0

Bayer_Matrix: size 8
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
```

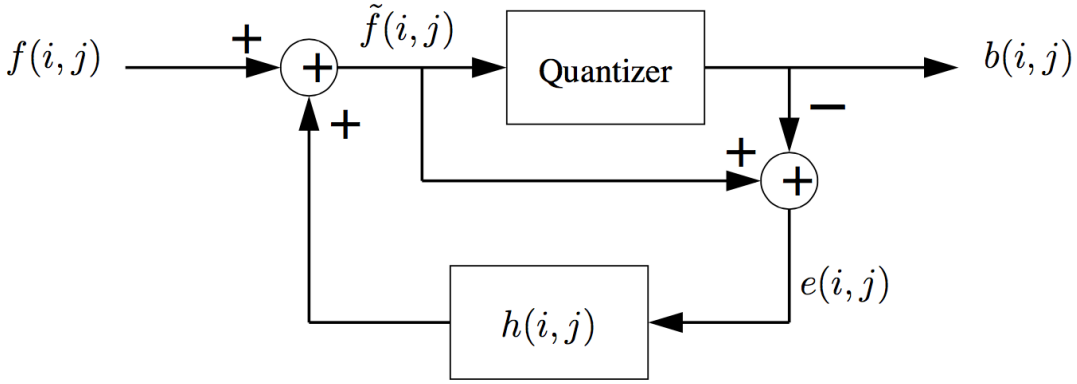
```
Bayer_Matrix: size 2
95.625 159.375
223.125 31.875

Bayer_Matrix: size 4
87.6562 151.406 103.594 167.344
215.156 23.9062 231.094 39.8438
119.531 183.281 71.7188 135.469
247.031 55.7812 199.219 7.96875

Bayer_Matrix: size 8
85.6641 149.414 101.602 165.352 89.6484 153.398 105.586 169.336
213.164 21.9141 229.102 37.8516 217.148 25.8984 233.086 41.8359
117.539 181.289 69.7266 133.477 121.523 185.273 73.7109 137.461
245.039 53.7891 197.227 5.97656 249.023 57.7734 201.211 9.96094
93.6328 157.383 109.57 173.32 81.6797 145.43 97.6172 161.367
221.133 29.8828 237.07 45.8203 209.18 17.9297 225.117 33.8672
125.508 189.258 77.6953 141.445 113.555 177.305 65.7422 129.492
253.008 61.7578 205.195 13.9453 241.055 49.8047 193.242 1.99219
```

Figure 14. Left: I2, I4, I8 matrix. Right: T2, T4, T8 matrix

2.2.2 Error Diffusion



- Equations are

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

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

$$\tilde{f}(i,j) = f(i,j) + \sum_{k,l \in S} h(k,l)e(i-k, j-l)$$

- Parameters
  - Threshold is typically $T = 127$.
  - $h(k,l)$ are typically chosen to be positive and sum to 1

Figure 15. Workflow and formula of Error Diffusion

The workflow and formula is in Figure 15[4]. First, initialized f matrix, for each pixel (from left to right, top to bottom), set b(i, j). Then get e(i, j) = f(i, j) - b(i, j) and diffuse error e forward to next pixels (depend on the size of E matrix).

There are three E matrix used in this problem.

$$Floyd-Steinberg's\ error\ diffusion\quad \frac{1}{16}\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 7\\ 3 & 5 & 1\end{bmatrix}$$

$$JJN\ error\ diffusion\quad \frac{1}{48}\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 7 & 5\\ 3 & 5 & 7 & 5 & 3\\ 1 & 3 & 5 & 3 & 1\end{bmatrix}$$

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

2.2.3 Scalar Color Halftoning

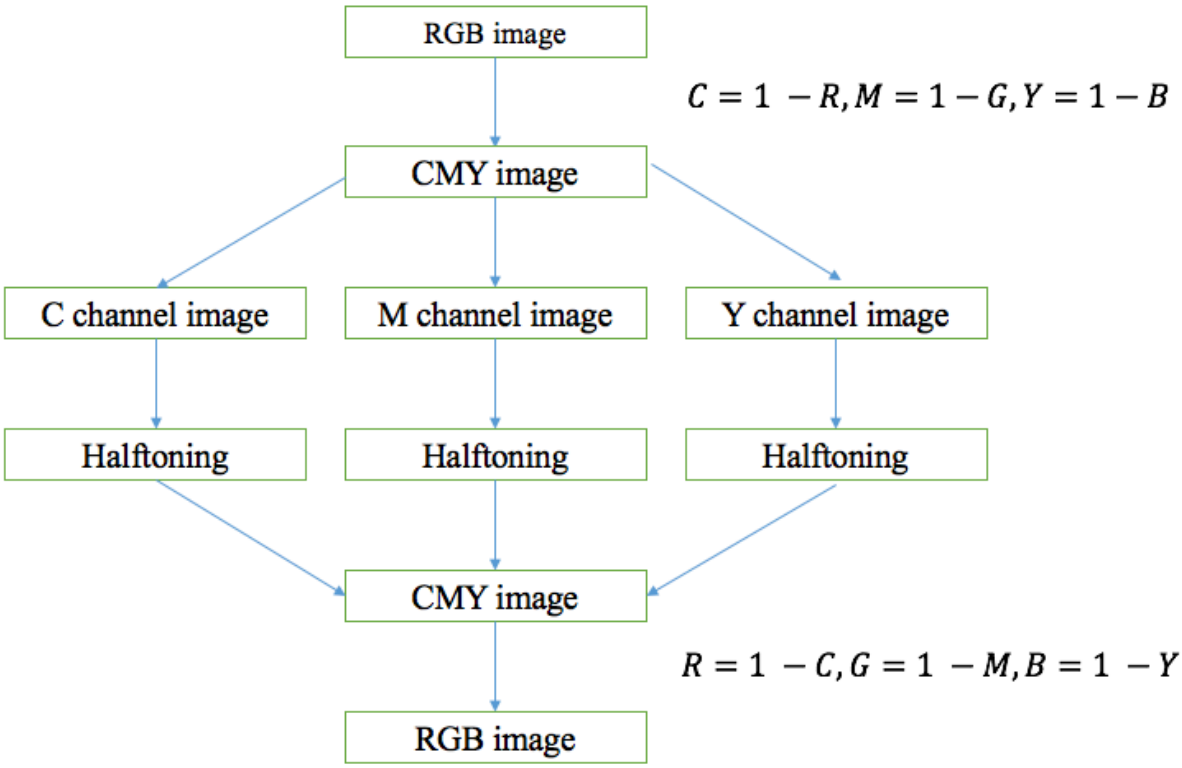


Figure 16. Workflow for Scale Color Halftoning

Color image halftoning is more complicated than grayscale image. One easy way of color halftoning is scalar color halftoning. Figure 16 is the workflow for this algorithm. Just separate image to three grayscale image and apply error diffusion halftoning to each image.

2.2.4 Vector Color Halftoning

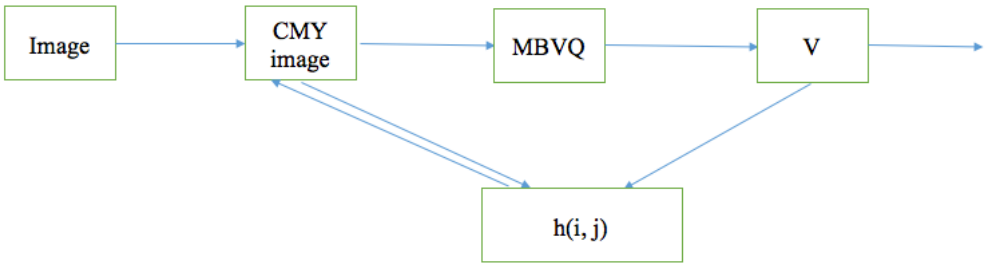


Figure 17. Workflow for Vector Color Halftoning

Vector color halftoning combine three color value together to generate halftoning image. The algorithm of Vector Color Halftoning is:

Initialize CMY(i, j) image, set e(i, j) = 0.

For each pixel: Find MBVQ(CMY(i, j), then find closest vector V in MBVQ to CMY(i, j) + e(i, j). Then set new e(i, j) = (CMY(i, j) + e(i, j) - V). Forward error e(i, j) by F-S Error Diffusion.

CMY cube. The eight vertexes of CMY cube are

$$W = (0,0,0), \quad Y = (0,0,1), \quad C = (0,1,0), \quad M = (1,0,0),$$
$$G = (0,1,1), \quad R = (1,0,1), \quad B = (1,1,0), \quad K = (1,1,1),$$

The algorithm of MBVQ modified by CMY image. This algorithm is referenced from [5].

```
if ((C + M) < 1)
    if ((C + M) < 1)
        if ((C + M + Y < 1)
            return CMYW;
        else
            return MYGC;
    else
        return RGMY;
else
    if ((M + Y) >= 1)
        if ((C + M + Y) >= 2)
            return KRGB;
        else
            return RGBM;
    else
        return CMGB;
```

The distance of V and CMV(i, j) is calculated by

$$distance = \sqrt{(C - V1)^2 + (M - V2)^2 + (Y - V3)^2}$$

2.3 Results

2.3.1 Dithering

Figure 18 is the dithering result of I2, I4, I8, the color intensities are 0 and 255. Figure 19 change the color intensities to 0, 85, 170, 255 four gray-levels.

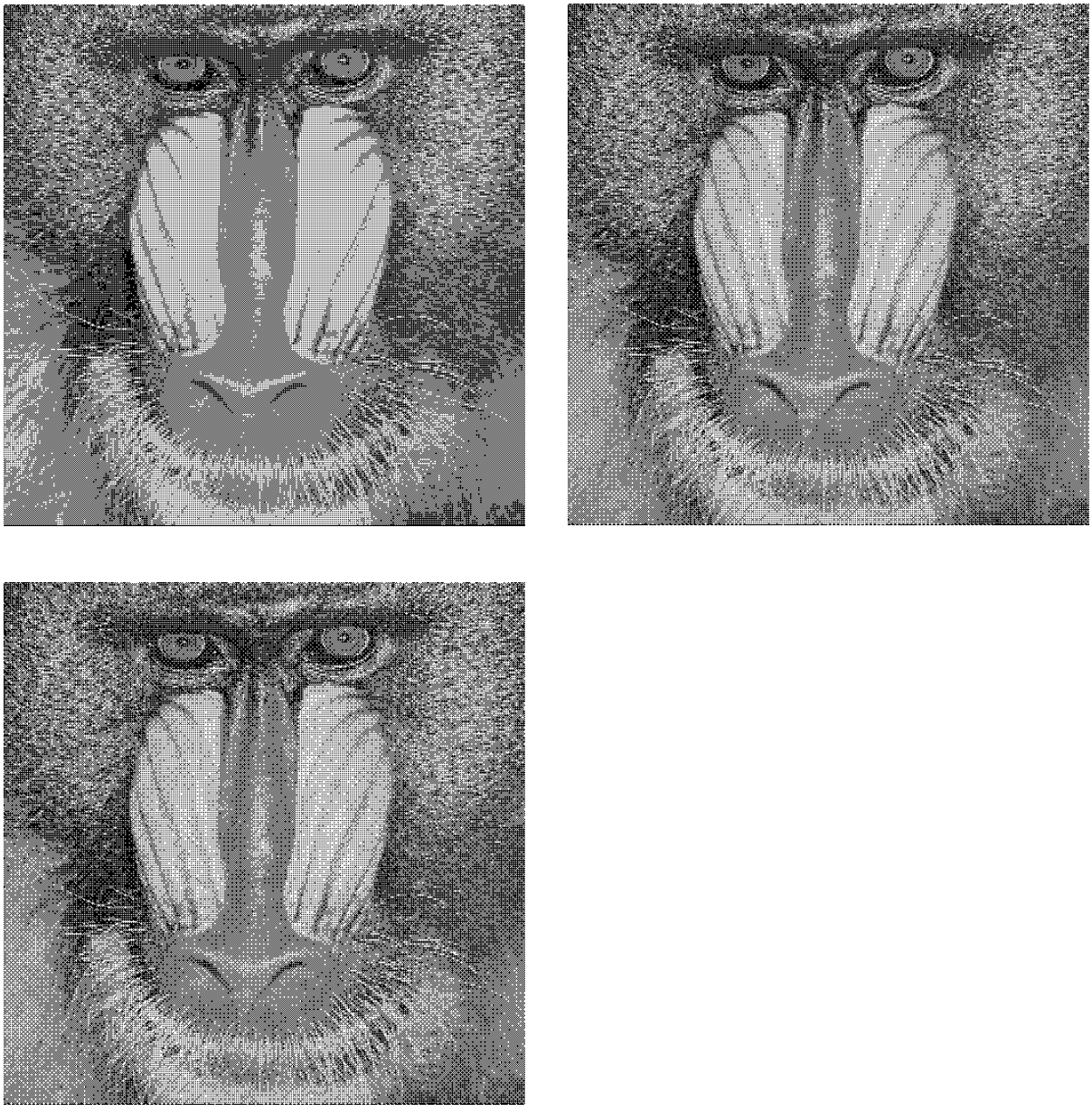


Figure 18. Dithering. Color (0, 255). Left to Right I2, I4, I8

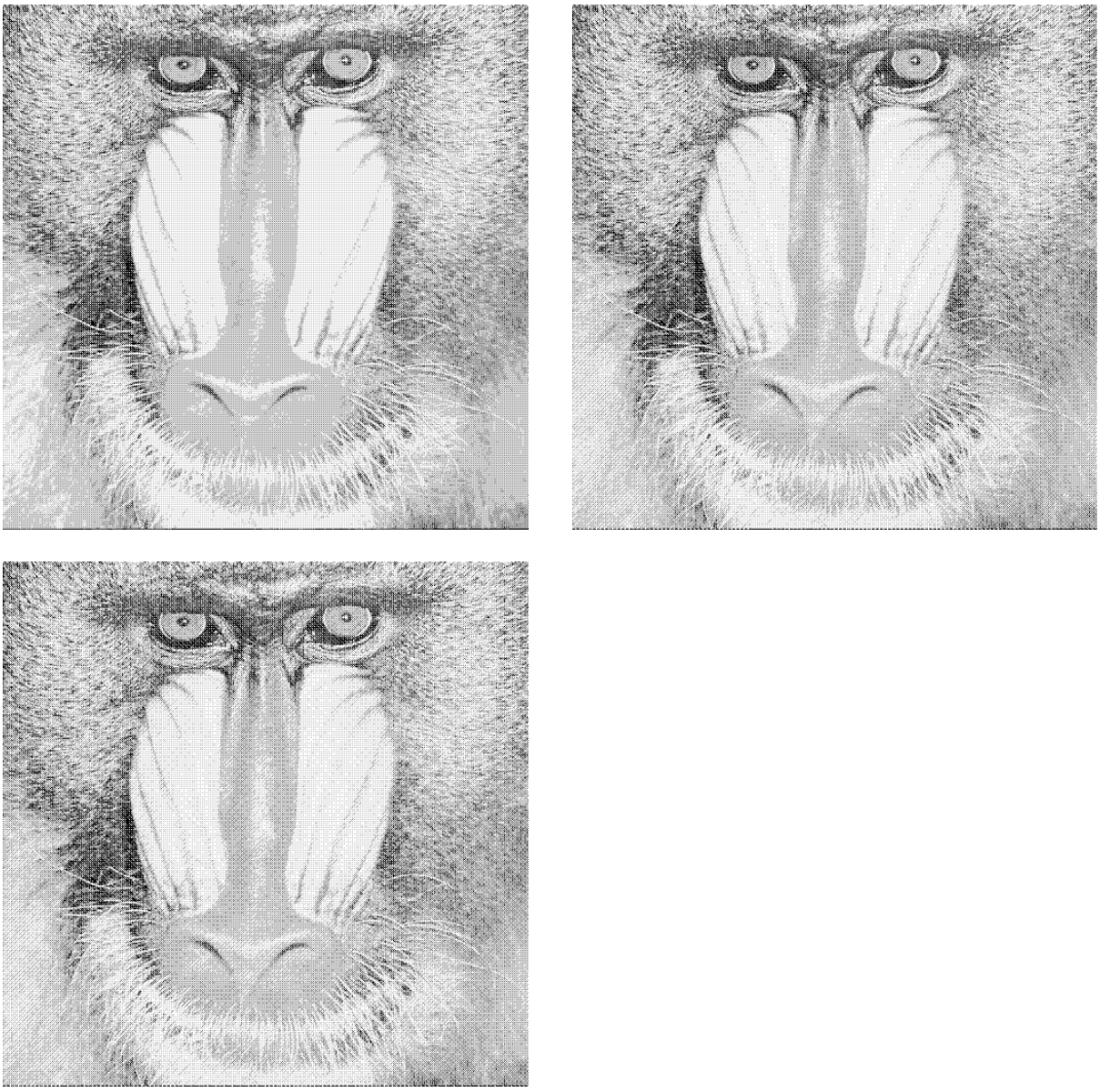


Figure 19. Dithering. Color (0, 85, 170, 255). Left to Right I2, I4, I8

2.3.2 Error Diffusion

Figure 20 is the results of error diffusion.

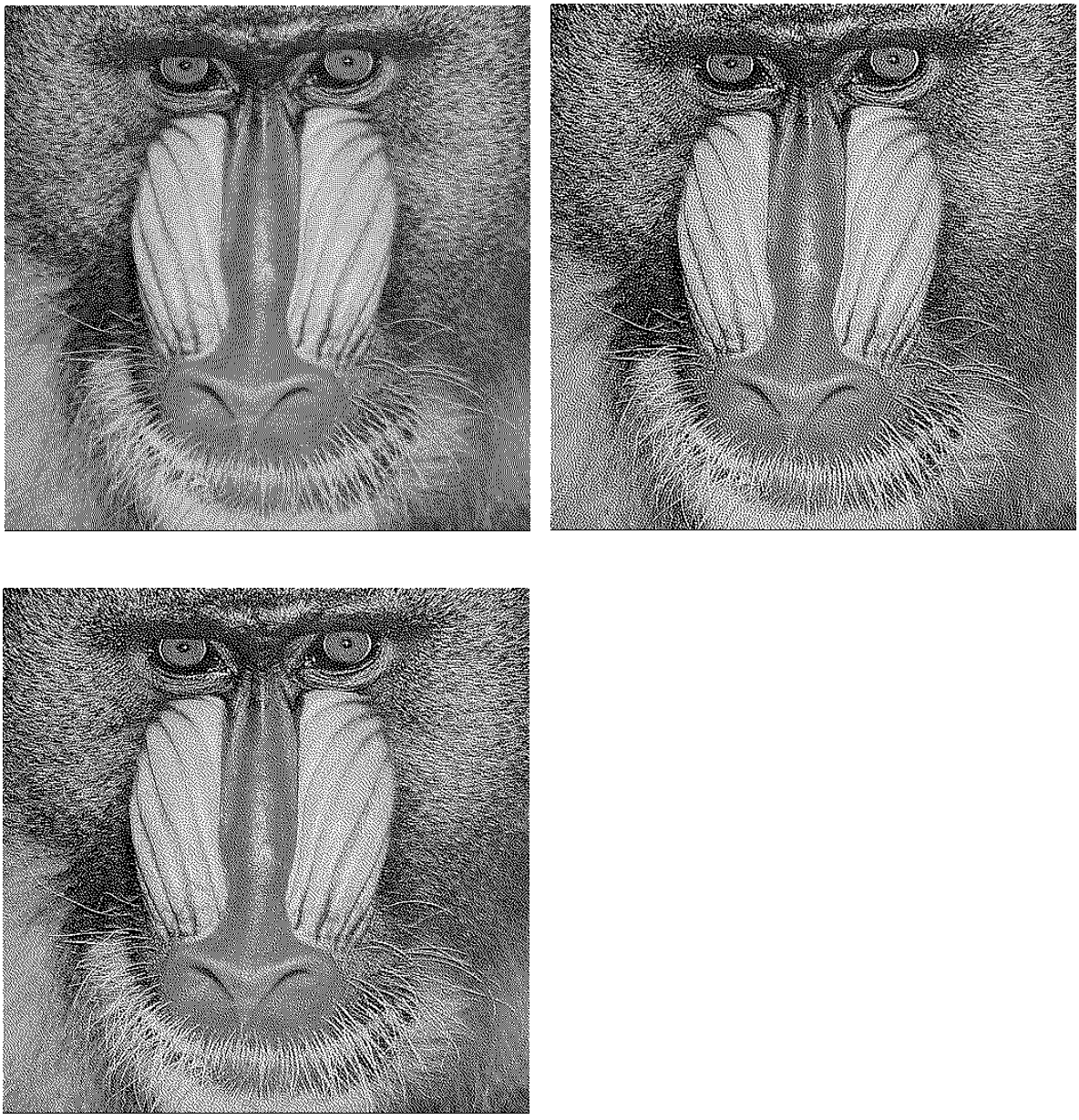


Figure 20. Error Diffusion. Left-Top: Floyd-Steinberg's error diffusion. Right-Top: Jarvis, Judice, and Ninke (JJN) error diffusion. Left-Bottom: Stucki error diffusion.

2.3.3 Scalar Color Halftoning

Figure 21 is the results of Scalar Color Halftoning.

Figure 21. Scalar Color Halftoning. Floyd-Steinberg's error diffusion

2.3.4 Vector Color Halftoning

Figure 22 is the results of Scalar Color Halftoning.

Figure 22. Vector Color Halftoning. Floyd-Steinberg's error diffusion

2.4 Discussion

2.3.1 Dithering

Dithering. Compared different index matrix I, I4 and I8 image results are similar, both better than I2. (Figure18)

Multi-grayscale. Here I use I matrix as 1/4 of I2, I4, I8, and apply dithering in each 85 range of intensity. Then add the results together.

2.3.2 Error Diffusion

Compared to the dithering, error diffusion half-toning gets a better result. The reason of this is that error diffusion use neighbor of pixels to evaluate the result. In this case, I4 and I8 index matrix can get a good result. We can see the texture shown in error diffusion is clearly.

2.3.3 Scalar Color Halftoning

Compared to the grayscale image, scalar color halftoning get a noisier image. This algorithm doesn't take other color in RGB into account.

2.3.4 Vector Color Halftoning

The difference between scalar color halftoning and vector color halftoning is that scalar color halftoning using other color of the same pixel to evaluate the result.

Why use MBVQ? Please see Figure 23. Left image is a threshold halftoning result with error diffusion and MBVQ. Right image is a threshold halftoning result without error diffusion but use MBVQ. We can see that MBVQ preserve more details on the texture of the image.

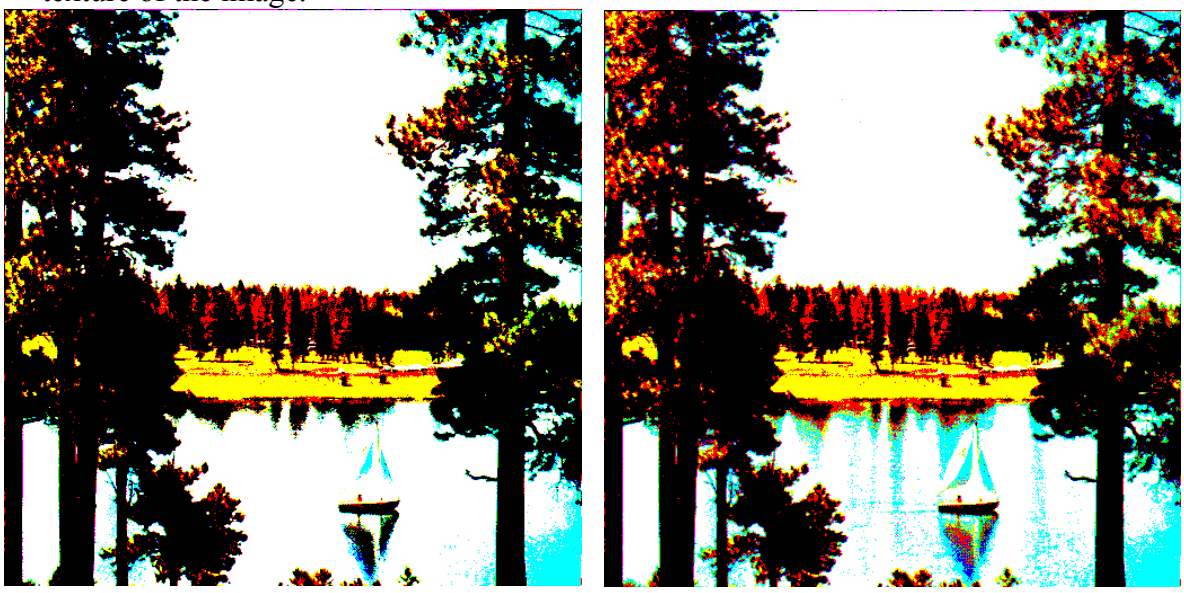


Figure 23. MBVQ

Problem 3. Morphological Processing

3.1 Motivation

Morphological image processing is a technologic to modify the spatial structure of objects in a digital image. Erosion, dilation, and skeletonization are three basic morphological operations. In the problem a, shrinking is implemented to erode the image to get the connectivity, objects information of image. In the problem b, thinning and skeletonizing are implemented to a one-object image to get the skeleton of the object. [6]

3.2 Approach and procedure

3.2.1 Shrinking

Erosion and dilation. The mathematics expression of erosion and dilation is below.

$$Dilation \quad A \oplus B = \{z \mid (\hat{B})z \cap A \ominus \neq \emptyset\}$$

$$Erosion \quad A \ominus B = \{z \mid (B)z \cap A^C \neq \emptyset\}$$

Open and close. Open is an operation that first erode, then dilate the image. This combined operation can smooth boundary and break thin connection. Close is an operation that first dilate, then erode the image. The operation can smooth boundary and fill holes in the image.

Hit-or-miss filter. Hit or miss is a combined operation. Shrinking, thinning, and skeletonizing use Hit-or-miss filter to detect boundary and false-true candidate pixels to remove. Figure 24 is the implementation of hit-or-miss filter. Shrinking, thinning and skeletonizing use different filter1, and filter2 provided in the textbook [6].

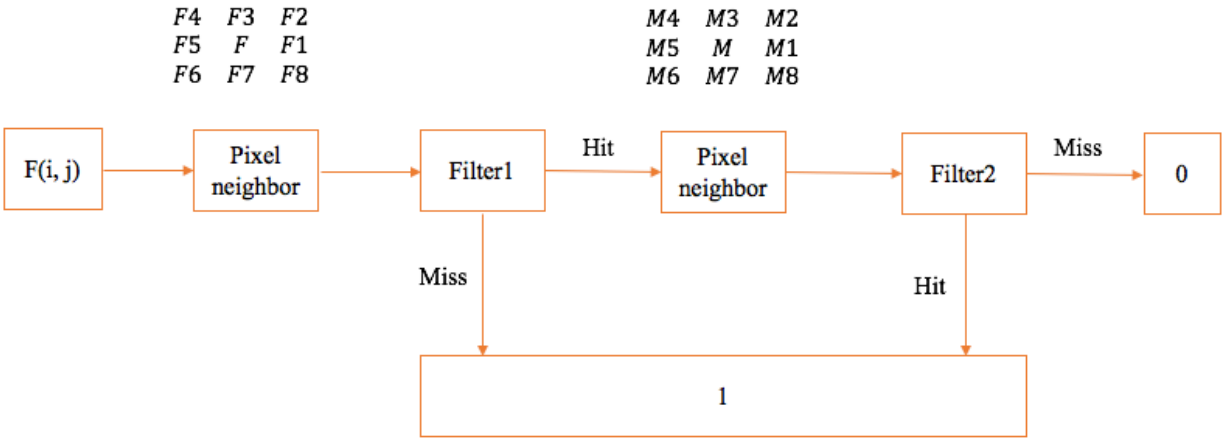


Figure 24. Hit-or-miss filter

Shrinking. Erode an object without holes to one single pixel. Erode an object without holes to a connected ring [6].

Connected component labeling. In problem a, we also use the technique of connected component labeling to count the number of disconnected pathway in the image. This algorithm scan the image by two passes and get the final result. The complexity of this algorithm is O(n^2). A label equivalence table is used store the labeling information of the image. The implementation of this algorithm is shown in Figure 25. [7]

- On the 1st pass:
  - Iterate through each element of the data by column, then by row (Raster Scanning)
  - If the pixel value is not the background
  - Get the neighboring non-background pixel of the current pixel
  - If there are no such neighbor, uniquely label the current pixel and continue
  - Otherwise, find the neighbor with the smallest label, and assign that label to current pixel
  - Store the equivalence between neighboring labels
- On the 2nd pass:
  - Iterate through each element of the data by column, then by row
  - If the pixel value is not the background: re-label the element with the lowest equivalent label

Figure 25. Algorithm of Connected component labeling [7]

3.2.2 Thinning and Skeletonizing

Thinning. Erode an object without holes to a connected stroke, and erode an object with holes to a connected ring.

Skeletonizing. Erode an object to its skeleton.

3.3 Results

3.3.1 Shrinking

Figure 26 is the result of finding nails. Figure 27 is the image in each step of operations. Totally, there are 8 nails in the image.

Figure 28 is the result of finding holes. Figure 29 is the image in each step of operations. As shown in the figure, there are 3 holes in the image.

Figure 30 is the result of finding objects. Figure 31 is the image in each step of operations. There are 8 pathways in the image.

```
Homework 3.3
Problem 3a
Find Nail
Operator_Erode: 0 1 0 1 1 1 0 1 0
Operator_Hit_Miss (Shrinking)
   COUNT: 462 ROUND: 6
Operator_Filter: 0 0 0 0 1 0 0 0 0

The image has 8 points
```

Figure 26. Result of finding nails.

Figure 27. Erode (0, 1, 0, 1, 1, 1, 0, 1, 0), Shrinking until convergence, Find-single point filter (0, 0, 0, 0, 1, 0, 0, 0, 0)

```
Homework 3.3
Problem 3a
Find Holes
Operator_Hit_Miss (Shrinking)
   COUNT: 733 ROUND: 65
Operator_Filter: 0 0 0 0 1 0 0 0 0

The image has 3 points
```

Figure 28. Result of finding holes.

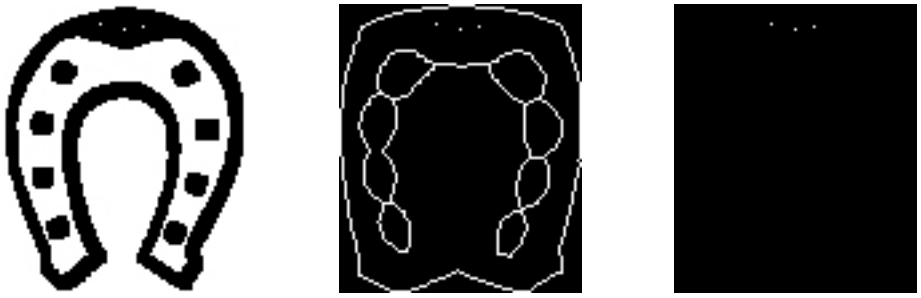


Figure 29. Turn black and white color. Shrinking until convergence, Find-single point filter (0, 0, 0, 0, 1, 0, 0, 0, 0)

```
Homework 3.3
Problem 3a
Find Objects
Operator_Erode: 1 1 1 1 1 1 1 1 1
Operator_Dilate: 0 1 0 1 1 1 0 1 0
Operator_Dilate: 0 1 0 0 1 0 0 0 0
Operator_Hit_Miss (Shrinking)
   COUNT: 425 ROUND: 10
The image has 9 pathways
```

Figure 30. Result of finding object

Figure 31. Left: Preprocess the image: erode the image by filter (1, 1, 1, 1, 1, 1, 1, 1, 1), dilate the image by filter (0, 1, 0, 1, 1, 1, 0, 1, 0), dilate the image by filter (0, 1, 0, 0, 1, 0, 0, 0, 0). Middle: Shrinking until convergence. Right: label the connectivity of the image.

3.3.2 Thinning and Skeletonizing

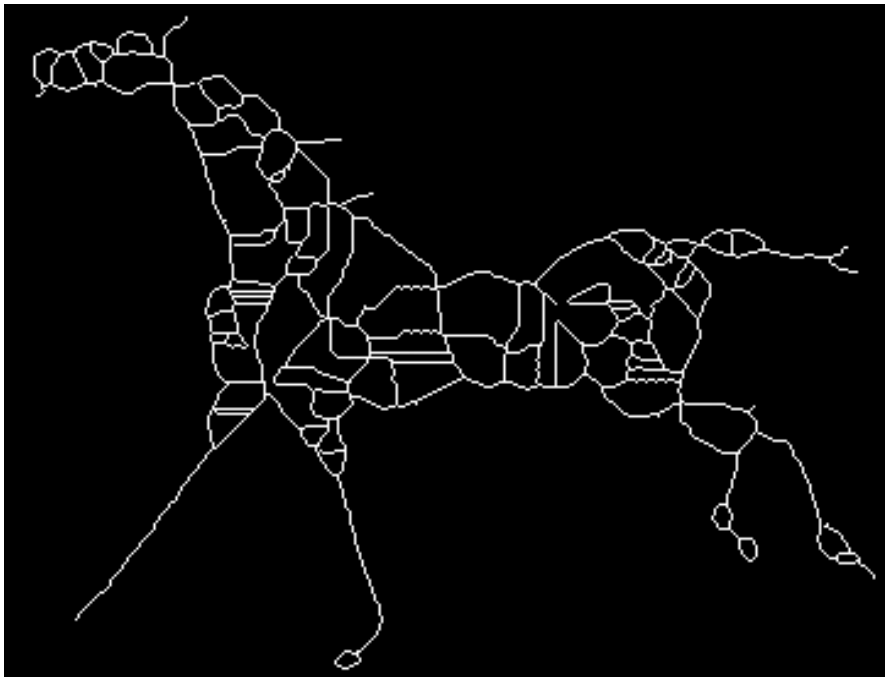


Figure 32. Thinning effect without preprocessing

Figure 32, 33, 34, 35 is results comparing the no preprocessing and with-preprocessing image. Figure 32, 33 is thinning effect, and Figure 34, 35 is skeletonizing effect.

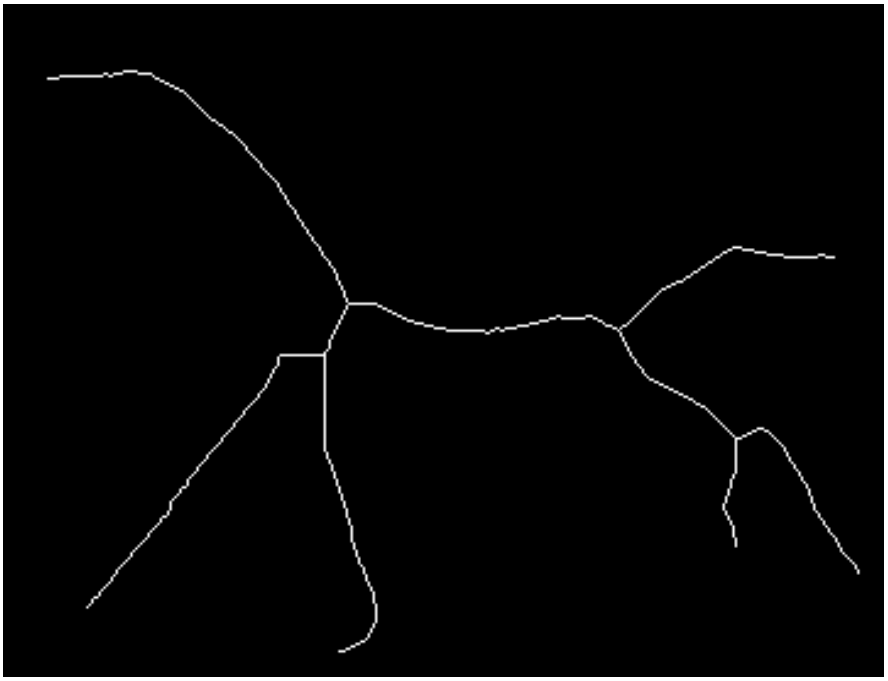


Figure 33. Thinning effect without preprocessing (2 close operations to smooth filter, 4 dilate operations to fill hole)

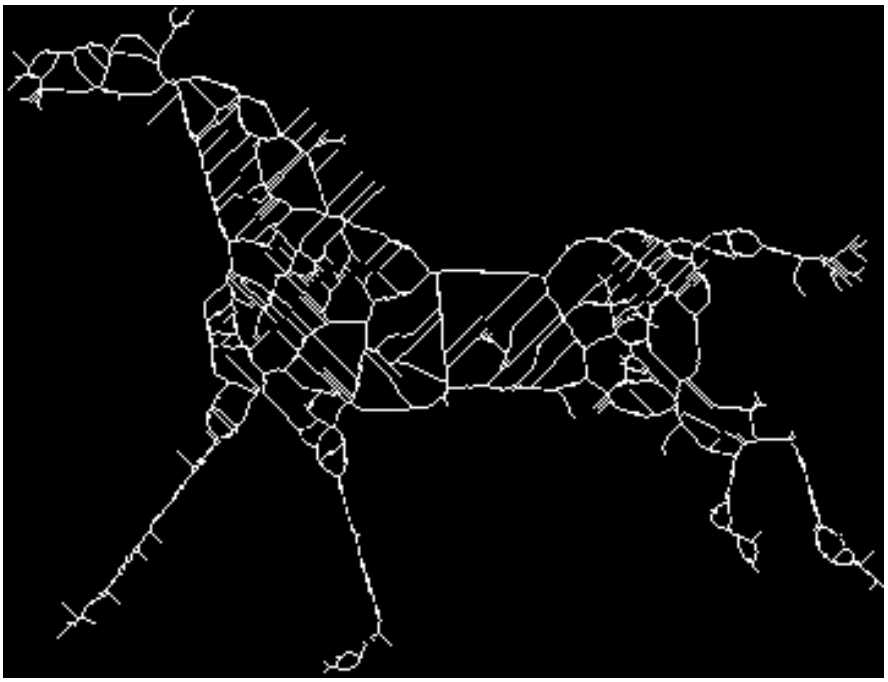


Figure 34. Thinning effect without preprocessing

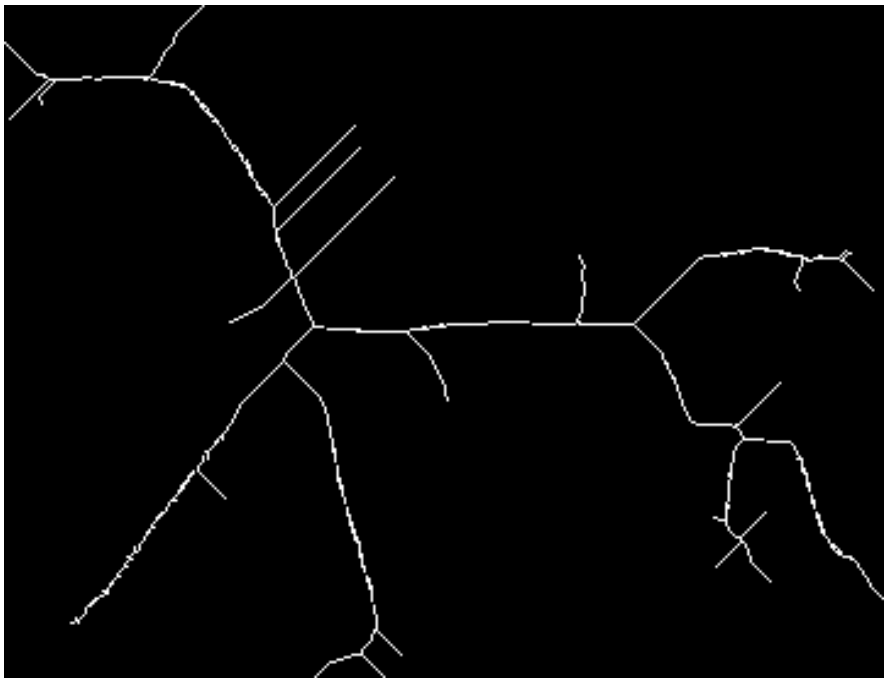


Figure 35. Thinning effect without preprocessing (2 close operations to smooth filter, 4 dilate operations to fill hole)

```
// Pre-processing
Dataset filter = {1, 1, 1, 1, 1, 1, 1, 1, 1};
Dataset filter2 = {0, 1, 0, 1, 1, 1, 0, 1, 0};
for (int i = 0; i < number_close; i++)
    op.Operator_Close(filter, filter2);
for (int i = 0; i < number_dilate; i++)
    op.Operator_Dilate(filter);

// Thinning
op.Operator_Hit_Miss(FILE_T1, FILE_T2, PATTERN_T1, PATTERN_T2);
```

Figure 36. Two filters for thinning, skeletonizing. Smoothing filter (1, 1, 1, 1, 1, 1, 1, 1, 1) (0, 1, 0, 1, 1, 1, 0, 1, 0), the first filter is used in close operation for dilation, and the second filter is used in close operation for erosion. Filling-hole filter (0, 1, 0, 1, 1, 1, 0, 1, 0)

3.4 Discussion

3.4.1 Shrinking

Shrinking. In problem a, shrinking is used in three counting method. We can see that after different pro-processing, shrinking can lead to different results. One of factor of choosing pre-processing filter is to see whether the counting object has a holes. And the counting object must be the white object, black is background.

Shrinking, thinning, and skeletonizing. The difference of shrinking, thinning and skeletonizing is shown in Figure 36. Shrinking is trying to erode the image to one point or ring, while thinning and skeletonizing erodes the image to thin line. We can see these three methods get a similar result (connected ring) in object with hole.

Figure 36. Left: Shrinking. Middle: Thinning. Right: Skeletonizing

3.4.2 Thinning and Skeletonizing

Problem b generate thinning and skeletonizing effect images to *Horse1* image.

Pre-processing filter. As shown in 32 to 35, the images with pre-processing filter (hole-filling filter and the boundary smoothing filter) are apparently better than the images without pre-processing filter.

Hole-filling filter is used fill the hole in the image. Here I use 3*3 all-one filter to dilate the image. Smoothing boundary filter is used to smooth the boundary. I use a close operation effect which include two filters (Dilate filter: 1, 1, 1, 1, 1, 1, 1, 1, 1; Erode: 0, 1, 0, 1, 1, 1, 0, 1, 0). To discuss the importance of two filters, I use images using different number of filters. Figure 38, 39 are a comparison result for different number of hole-filling filter applying. And Figure 40, 41 are a comparison result for different number of smoothing boundary filter. The effects of two filters are shown in Figure 37.

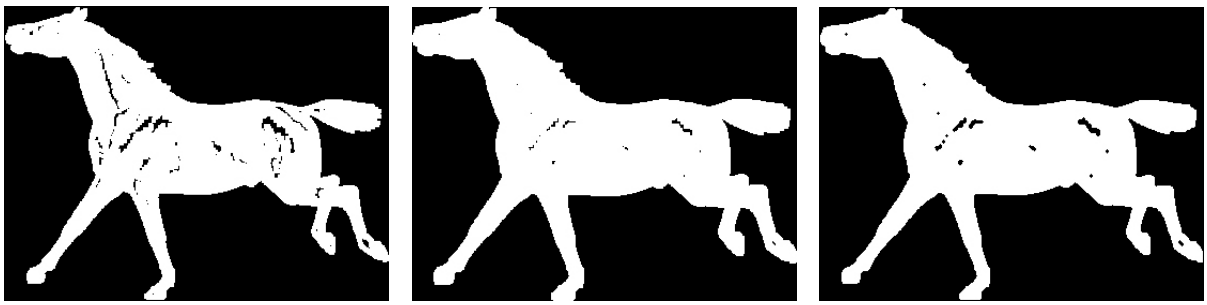


Figure 37. Left: Original image. Middle: Apply one hole-filling filter to the Horse. Right: apply one boundary smooth filter to the Horse

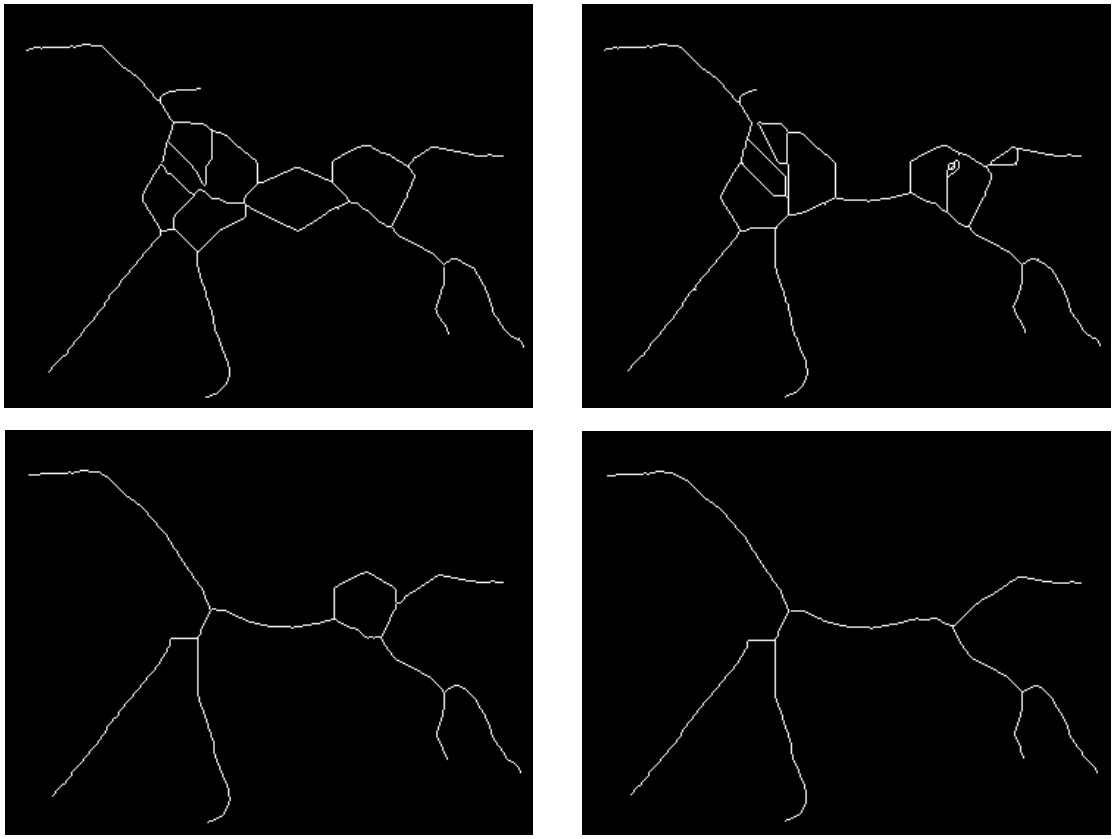


Figure 38. Thinning. Number of smoothing filter = 2. Top-left: Number of hole-filling filter = 0. Top-right: Number of hole-filling filter = 1. Bottom-left: Number of hole-filling filter = 2. Bottom-right: Number of hole-filling filter = 3

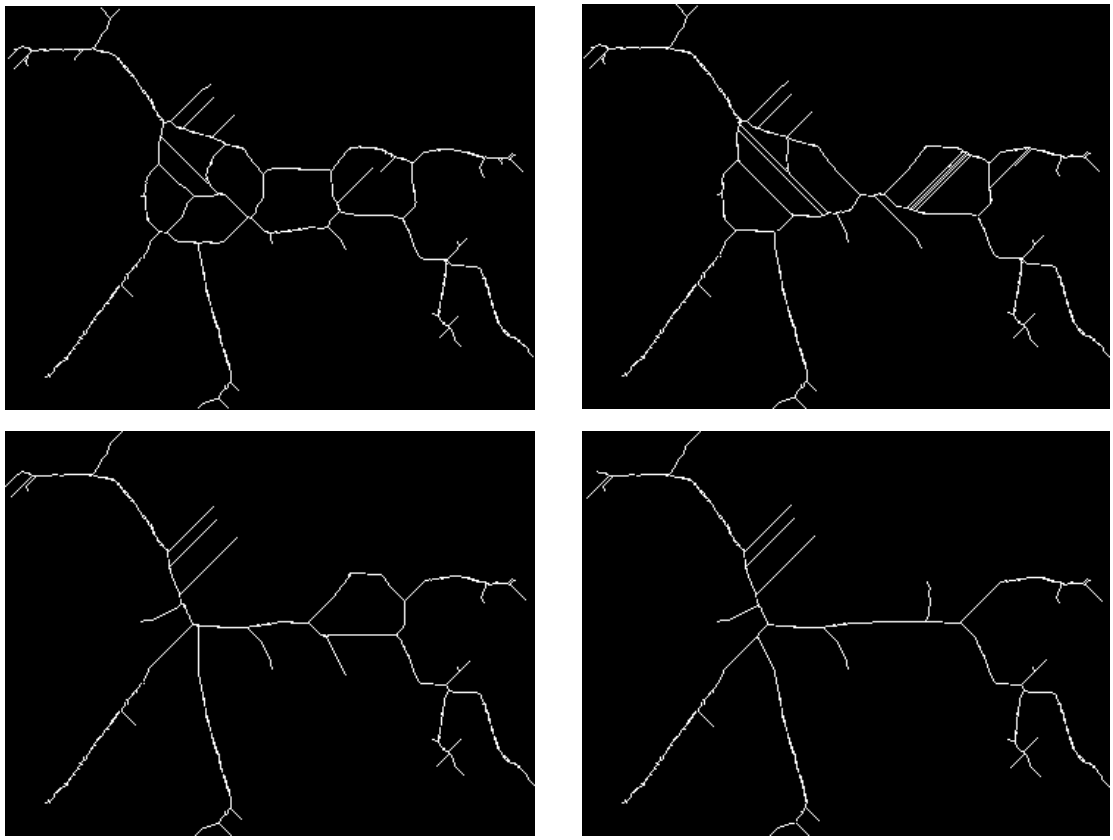


Figure 39. Skeletonizing. Number of smoothing filter = 2. Top-left: Number of

hole-filling filter = 0. Top-right: Number of hole-filling filter = 1. Bottom-left: Number of hole-filling filter = 2. Bottom-right: Number of hole-filling filter = 3

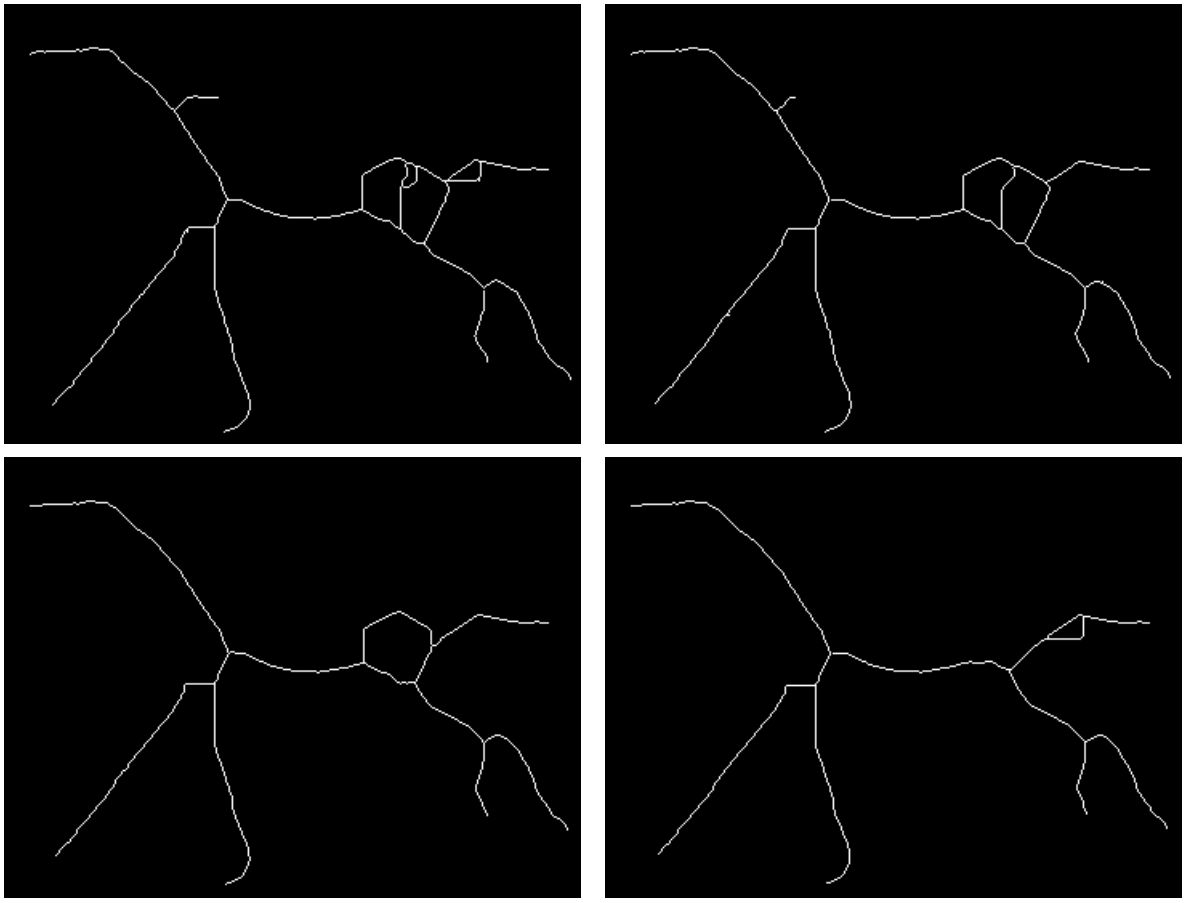


Figure 40. Thinning. Number of hole-filling filter = 2. Top-left: Number of smoothing filter = 0. Top-right: Number of smoothing filter = 1. Bottom-left: Number of smoothing filter = 2. Bottom-right: Number of smoothing filter = 3

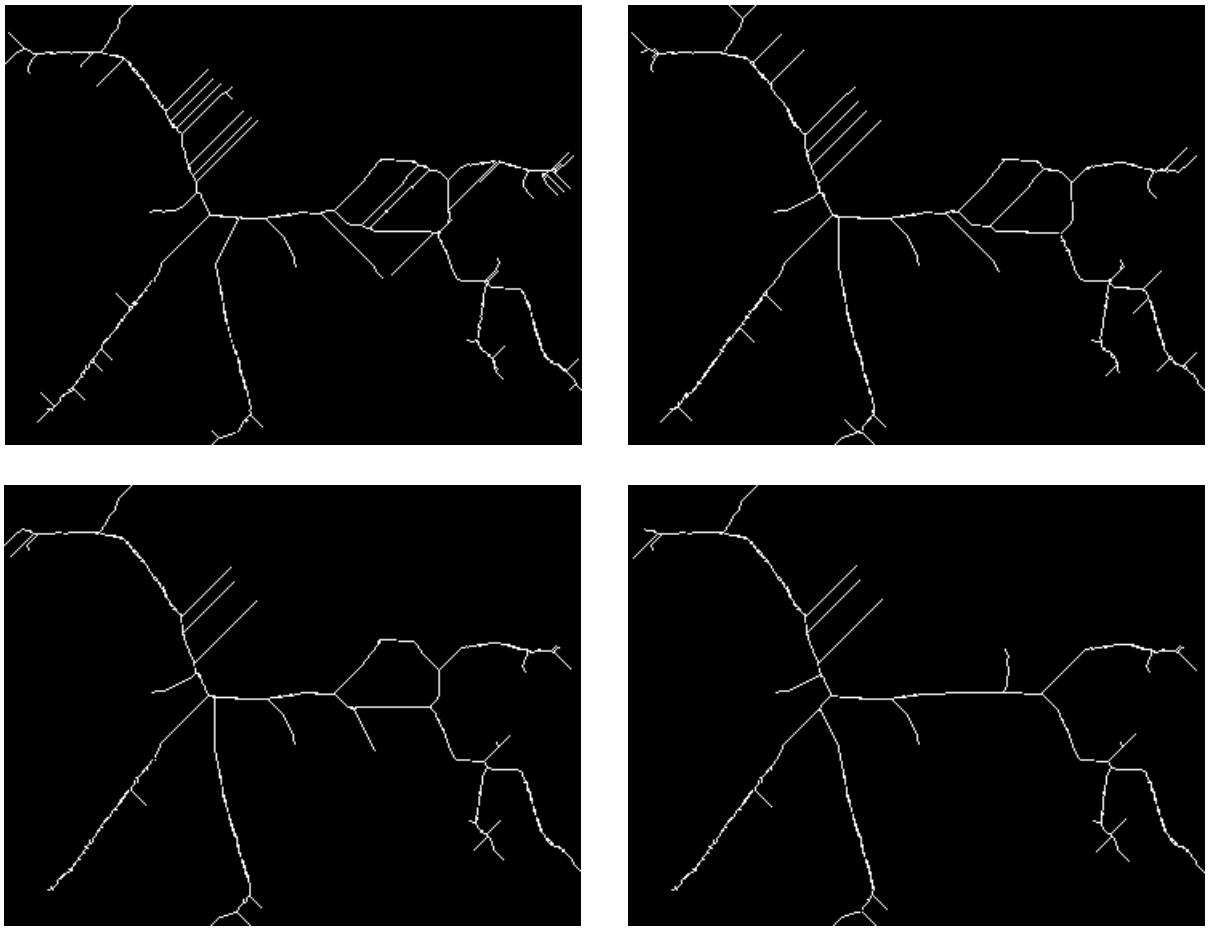


Figure 41. Skeletonizing. Number of hole-filling filter = 2. Top-left: Number of smoothing filter = 0. Top-right: Number of smoothing filter = 1. Bottom-left: Number of smoothing filter = 2. Bottom-right: Number of smoothing filter = 3

Compared Figure 38 to Figure 41, we can see smooth-boundary filter decrease the skeleton branch of the *Horse*, and the hole-filling filter decrease the ring result of the *Horse*.

Reference

[1] Spaces needed to coordinate modeling, animation, viewing, and rendering, CS580, Urich Neumann

[2] EE569 Homework 3

[3] https://en.wikipedia.org/wiki/Halftone

[4] C. A. Bouman: Digital Image Processing - January 12, 2015

[5] Shaked, Doron, et al. "Color diffusion: error diffusion for color halftones." Electronic Imaging'99. International Society for Optics and Photonics, 1998.

[6] Digital Image Processing, PIKS Scientific Inside. William K. Pratt

[7] EE569 Discussion 9, Chun-Ting Huang