EE 569: Homework #3

Issued: 10/9/2015 Due: 11:59PM, 11/1/2015

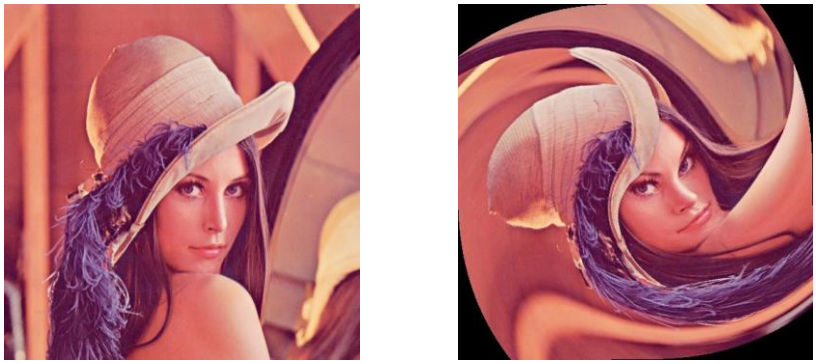
General Instructions:

1. Read Homework Guidelines and MATLAB Function Guidelines for the information about homework programming, write-up and submission. If you make any assumptions about a problem, please clearly state them in your report.
2. You need to understand the USC policy on academic integrity and penalties for cheating and plagiarism. These rules will be strictly enforced.

Problem 1: Geometrical Modification (40%)

(a) Swirl Effect (Basic: 10%)

You can create special visual effects using the geometric transformation. One example called the swirl effect is shown in Figure 1, which has the *Lena* image as the input. Write a program to implement the swirl special effects and apply it to the *Kate* image in Fig. 2. Describe your method and present the result in your report. (Hint: you need to consider swirling and rotation.)



(a) Original Lena (b) Lena with the swirl effect

Figure 1: Illustration of the swirl effect

Figure 2: Kate.raw

(b) Perspective Transformation & Imaging Geometry (30%)

In this problem, you will use the perspective transformation and imaging geometry to capture a scene in the 3D world coordinates system and project it onto a 2D image plane.

A cube in the 3D world coordinates system is shown in Figure 3, where its center is located at the origin and each side is parallel to one of the XY-, XZ- and YZ-planes. One viewing camera has its lens centered at (5,5,5), viewing angle along the direction of (-1,-1,-1), and focal length *f*. Please show the image observed at the image plane in this problem.

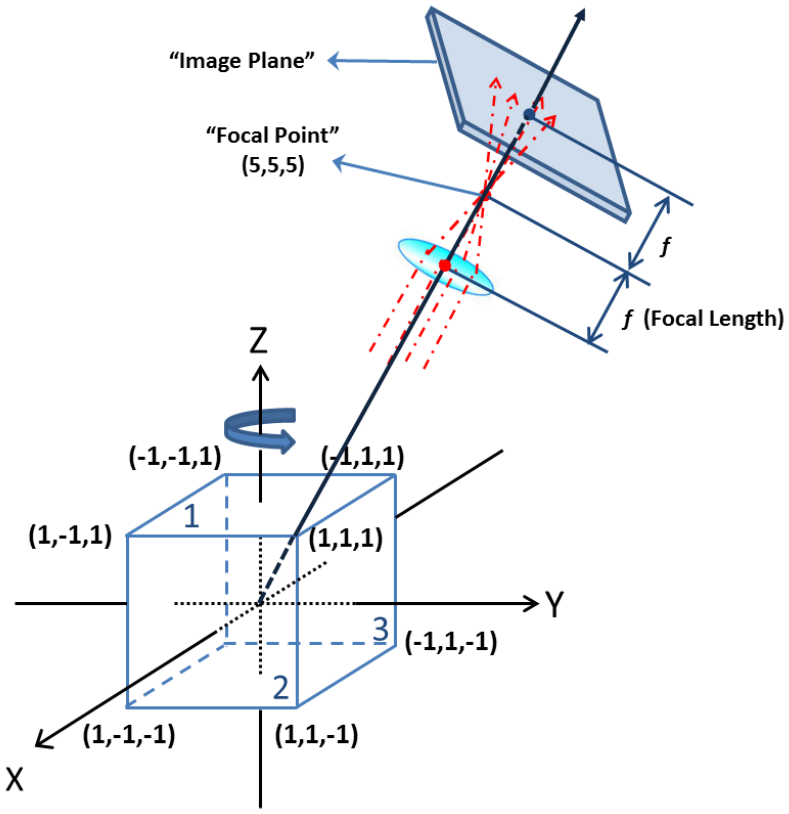


Figure 3: 3D Cube in the World Geometry

To conduct this task, you need to understand and implement two coordinates transform matrices as given below. Generally, the image plane coordinates, $[x, y]^T$, are related to the 3D world coordinates $[X, Y, Z]^T$ via

$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

where K and $[R|t]$ are called the **intrinsic** camera matrix and the **extrinsic** camera matrix, respectively, and w is a scaling factor. Matrices K and $[R|t]$ are given below.

1. **Extrinsic Camera Matrix: from World Coordinates to Camera Coordinates**
To map the location of a point in the world coordinates to that of the image plane, we need to change it to the camera coordinates system (see Figure 4) as an intermediate step, and $[R|t]$ in Eq. (1) is used to accomplish this task.

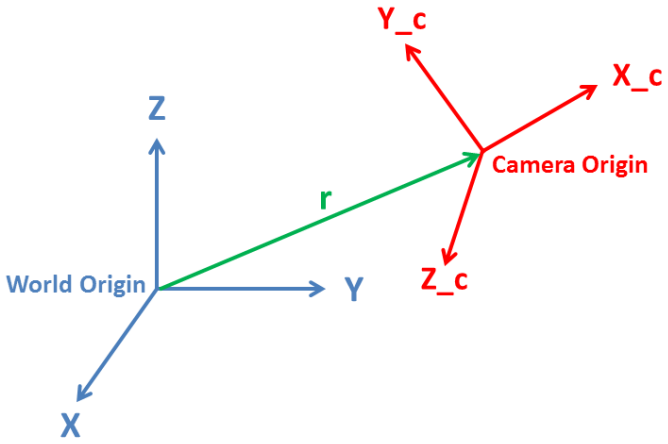


Figure 4: Relationship between the world coordinates and the camera coordinates.

It can be rewritten as

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} = \begin{bmatrix} X_c^X & X_c^Y & X_c^Z & -\mathbf{r}\cdot\mathbf{X_c} \\ Y_c^X & Y_c^Y & Y_c^Z & -\mathbf{r}\cdot\mathbf{Y_c} \\ Z_c^X & Z_c^Y & Z_c^Z & -\mathbf{r}\cdot\mathbf{Z_c} \end{bmatrix},$$

where $\mathbf{r}$ is the vector from the origin of the world coordinates to the origin of the camera coordinates, and $\mathbf{X}_c=(X_c^X, X_c^Y, X_c^Z)^T$, $\mathbf{Y}_c=(Y_c^X, Y_c^Y, Y_c^Z)^T$, $\mathbf{Z}_c=(Z_c^X, Z_c^Y, Z_c^Z)^T$ are the unit vectors of the camera coordinates system with respect to the world coordinates system, respectively. Based on the configuration in Figure 3, we have

$$\mathbf{Z_c} = (-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})^T,$$

since the image plane faces the origin of the world coordinates and $\mathbf{r}$=(5,5,5). Without loss of generality, we choose the following initial values for $\mathbf{X}_c$ and $\mathbf{Y}_c$ in matrix $[R|t]$:

$$\mathbf{X_c} = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)^T, \qquad \mathbf{Y_c} = (\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, -\frac{2}{\sqrt{6}})^T.$$

2. **Intrinsic Camera Matrix: from Camera Coordinates to Image Plane Coordinates**
Next, we map the location of a point in the camera coordinates to that of the image plane. In Figure 5, X_c, Y_c and Z_c are three axes of the camera coordinates, and **P** and **P'** are points in the camera coordinates and its projective point onto the image plane, respectively. Matrix K in Eq. (1) can be derived using the perspective image transformation as

$$\omega \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} f \cdot X_c + c_x \cdot Z_c \\ f \cdot Y_c + c_y \cdot Z_c \\ Z_c \end{bmatrix}$$

where (c_x, c_y) is the image coordinates of the intersecting point between the optical axis and the image plane.
After the simplification, we have

$$x = \frac{f \cdot X_c + c_x \cdot Z_c}{Z_c} = f \cdot \frac{X_c}{Z_c} + c_x, \qquad y = \frac{f \cdot Y_c + c_y \cdot Z_c}{Z_c} = f \cdot \frac{Y_c}{Z_c} + c_y.$$

Note that $c_x = \frac{ImageWidth}{2}$ and $c_y = \frac{ImageHeight}{2}$ based on the image coordinates convention. Also, by shifting the origin of the image plane to (c_x, c_y), one can verify that

$$x : X_c = f : Z_c \leftrightarrow x = f \cdot \frac{X_c}{Z_c}, \qquad y : Y_c = f : Z_c \leftrightarrow y = f \cdot \frac{Y_c}{Z_c}$$

as shown in Figure 5.

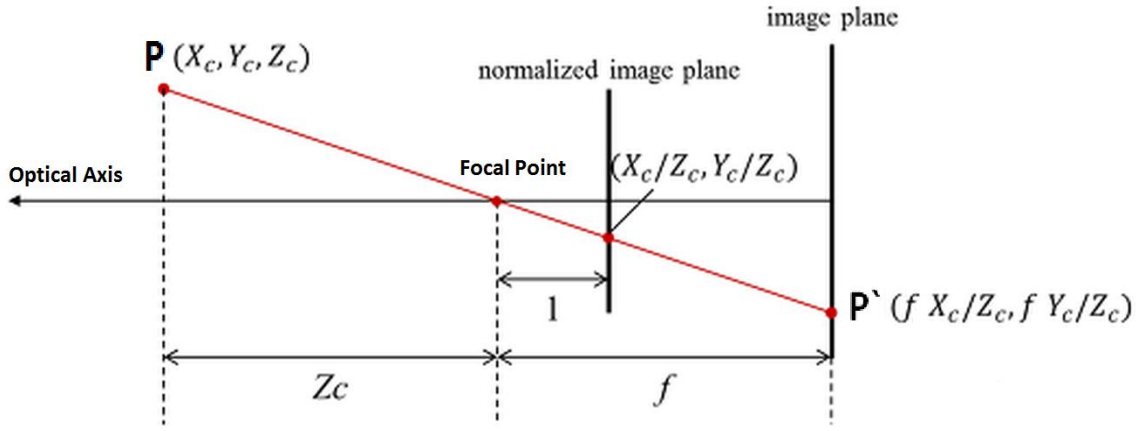


Figure 5: Relationship between the camera and the image plane coordinates.

i. Pre-processing (Basic: 10%)
Figure 6 shows five 200x200 images which will be placed on five faces of the cube as shown in 6(f). You can ignore the bottom face in this problem. With the configuration in Figure 3, the camera can observe images on face no.1, no. 2 and no. 3.

Write a program to generate the location/intensity information of the five images by assuming the unit length of the world coordinates is 0.01. Because the resolution of the image is 200x200 and the side length of the cube is equal to 2, we have $\frac{2}{200} = 0.01$. Explain the basic idea of your implementation. Please specify the input and the output of your program clearly.

ii. Capturing 3D scene (Advanced: 20%)

In this part, you need to display the cube image on the image plane captured by the camera. By following the aforementioned steps, you can determine K and $[R|t]$.

Please first implement the forward mapping algorithm (from the world coordinates to the camera coordinates) with the following three fixed parameters:

- Focal length $f = \sqrt{3}$,
- Image width w = 200 pixels,
- Image height h = 200 pixels.

There is one more parameter called the pixel density (namely, the number of pixels per unit length) for you to determine. Note that a poor choice of pixel density will result in a projected image either too small or too big. Please specify one good value of pixel density and provide the reason for your choice. Discuss the resulting projected image. If there are undesirable artifacts, explain the source of the distortions. Is there any way to improve the result? If so, describe your own algorithm (you do not need to actually implement it).

Then, with the set of parameters obtained above, please implement the reverse mapping (from the camera coordinates to the world coordinates). What are the challenges in the reverse mapping?

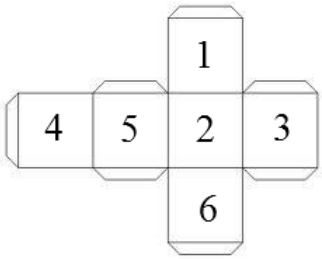
(a) baby

(b) baby cat

(c) baby dog

(d) baby panda

(e) baby bear



(f) Unfolded cube image

Figure 6: Mapping five images to five faces of the cube.

Problem 2: Digital Halftoning (30%)

(a) Dithering (Basic: 6%)

Convert the 8-bit *Mandrill* image in Fig. 7 to a half-toned image using the dithering method.

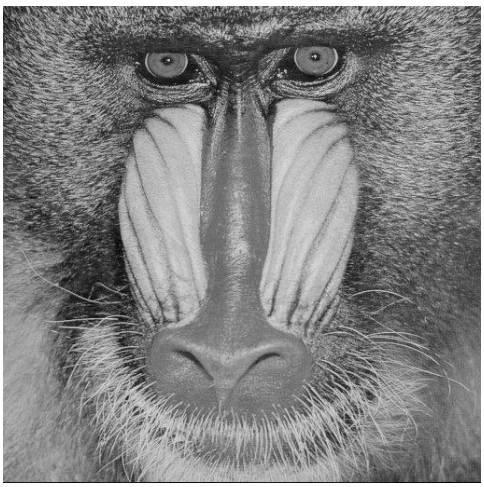


Figure 7: mandrill.raw

Dithering parameters are specified by an index matrix. The values in an index matrix indicate how likely a dot will be turned on. For example, an index matrix is given by

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

where 0 indicates the pixel most likely to be turned on, and 3 is the least likely one. This index matrix is a special case of a family of dithering matrices first introduced by Bayer [1].

The Bayer index matrices are defined recursively using the formula:

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

The index matrix can then be transformed into a threshold matrix T for an input gray-level image with normalized pixel values (*i.e.* with its dynamic range between 0 and 1) by the following formula:

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

where N^2 denotes the number of pixels in the matrix. Since the image is usually much larger than the threshold matrix, the matrix is repeated periodically across the full image. This is done by using the following formula:

$$G(i,j) = \begin{cases} 1 & \text{if } F(i,j) > T(i \bmod N, j \bmod N) \\ 0 & otherwise \end{cases}$$

where *F(i,j)* and *G(i, j)* are the normalized input and output images. Answer the following questions.

1. Construct $I_2(i,j)$ and $I_8(i,j)$ Bayer index matrices and apply them to the *Mandrill* image.
2. If a screen can only display FOUR intensity levels, design a method to generate a display-ready *Mandrill* image. Show your best result in gray-scale with four gray-levels (0, 85, 170, 255) and explain your design idea and detailed algorithm.

(b) Error Diffusion (Basic: 6%)

Convert the 8-bit *Mandrill* image to a half-toned one using the error diffusion method. Show the outputs of the following three variations, and discuss these obtained results.

1. Floyd-Steinberg's error diffusion with the serpentine scanning, where the error diffusion matrix is
$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix}.$$
2. Error diffusion proposed by Jarvis, Judice, and Ninke (JJN), where the error diffusion matrix is
$$\frac{1}{48}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$
3. Error diffusion proposed by Stucki, where the error diffusion matrix is
$$\frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

Describe your own idea to get better results. There is no need to implement it if you do not have time. However, please explain why your proposed method will lead to better results.

(c) Scalar Color Halftoning (Basic: 6%)

A naive idea is to apply the monochrome halftoning method independently to the colorant [cyan, magenta, yellow, and black (CMYK)] planes. Implement this idea as follows: Separate an image into CMY three channels and apply the Floyd-Steinberg error diffusion algorithm to quantize each channel separately. Then, you will have one of the following 8 colors, which correspond to the 8 vertices of the CMY cube at each pixel:

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

Note that (W, K), (Y, B), (C, R), (M, G) are complementary color pairs as illustrated in Figure 5. Please show and discuss the result of the half-toned color *Sailboat* image. What is the shortcoming of this approach?

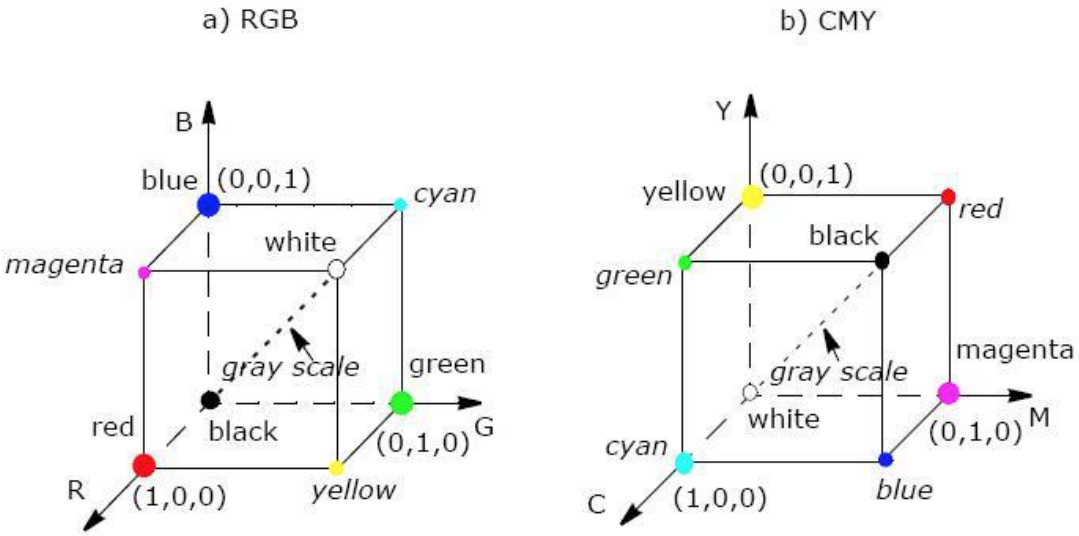


Figure 8: RGB and CMY Cube

(d) Vector Color Halftoning (Advanced: 12%)

We expect better results by conducting color halftoning in three channels jointly. It is called the "vector color halftoning" technique. Shakad *et al*. [2] proposed a vector color error diffusion method. They partition the CMY color space into 6 Minimum Brightness Variation Quadruples (MBVQ) as shown in Fig. 2 of [2]. Let the CMY value and its quantization error at pixel (x, y) be CMY(x, y) and e(x, y), respectively. The error diffusion can be conducted as follows:

Initialization: Set the quantization error at all pixels to zero.
MBVQ Quantization and Color Error Diffusion: Scan pixels in the input image with the serpentine order and repeat the following operations.

1) **MBVQ Quantization:** Find the vertex, V, of the MBVQ tetrahedron that is closest to CMY(x, y) + e(x, y) and output the value of V at location (x, y);
2) **Color Error Diffusion:** Compute the new quantization error using CMY(x, y) + e(x, y) – V, and distribute the quantized error to the error buckets e(x,y) of future pixels using a standard error diffusion process (e.g. the FS error diffusion).

You may refer to [2] for more details. Implement this algorithm and apply it to the *Sailboat* image in Figure 6. Compare the output with that obtained in 1. . Comment on the differences between these two methods.

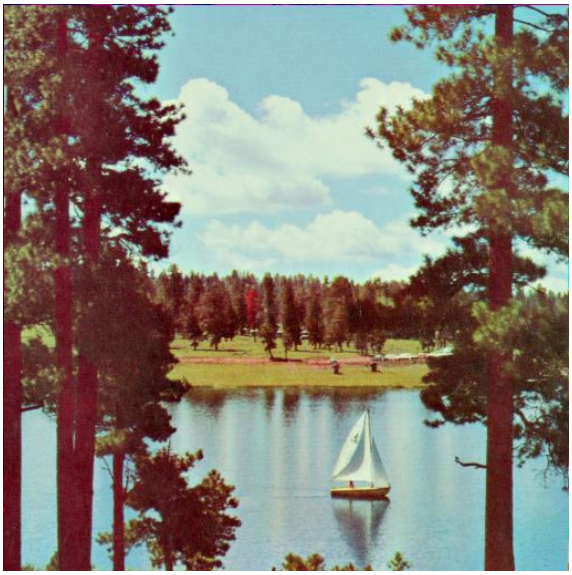


Figure 9: Sailboat.raw

Problem 3: Morphological Processing (30%)

Implement three morphological processing operations: shrinking, thinning, and skeletonizing. A pattern table (patterntables.pdf) is attached for your reference. Show outputs for all following parts in your report and discuss them thoroughly. State any assumptions you make in your solution. Images for this problem all come from MPEG-7 Shape Dataset [3].

(a) Shrinking (Basic: 10%)

Apply the shrinking filter to the horseshoe image as shown in Figure 7. Answer following questions:

1. How many nails (white circle) and holes (black dots) in the image? Use your program to derive the result.
2. How many white objects (includes horseshoe itself) in the image? Use your program to derive the result. (Hint: use hole-filling filter)

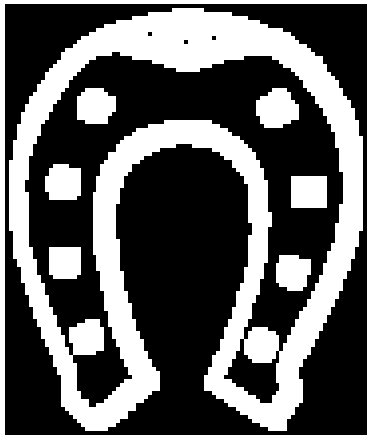


Figure 7: Horseshoe.raw

(b) Thinning and Skeletonizing (Basic: 10%)

First, apply the hole-filling filter and the boundary smoothing filter as the pre-processing operations to remove interior holes and smoothen the contour of the horse binary image in Figure 8. Next, perform the thinning and skeletonizing filters to the cleaned horse object. Compare the thinning and skeletonizing results with and without the pre-processing step and comment on the importance of pre-processing.

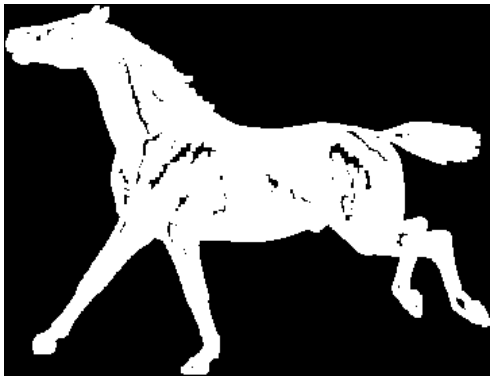
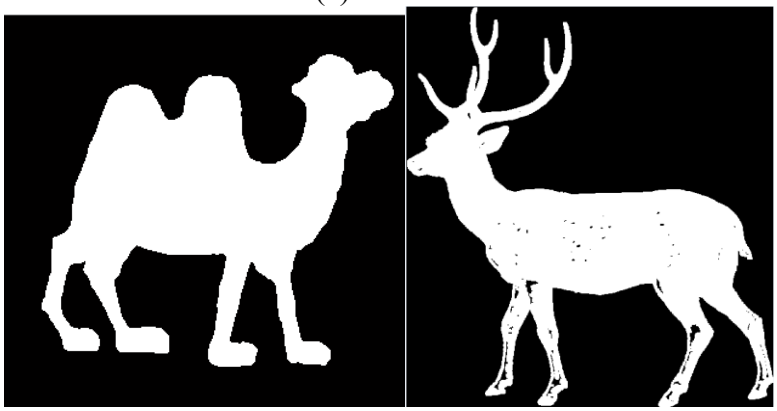


Figure 8: Horse1.raw

(c) Shape Recognition (Advanced: 10%)

Extract features from the thinned/skeletonized results of Horse2, Camel, and Deer images in Figure 9. Rank the similarity between Horse1 and other images (Horse2, Camel, and Deer) based on the selected feature distance. The one with the minimum distance represents the closest match. Discuss your observations and show the ranking of these three images.

(a) Horse2.raw



(b) Camel.raw (c) Deer.raw

Figure 9: Three animal shapes

Appendix:

Problem 1: Geometrical Modfications

lena.raw	512x512	24-bit	color(RGB)
lena_swirl.raw	512x512	24-bit	color(RGB)
kate.raw	512x512	24-bit	color(RGB)
baby.raw	200x200	24-bit	color(RGB)
baby_cat.raw	200x200	24-bit	color(RGB)
baby_dog.raw	200x200	24-bit	color(RGB)
baby_panda.raw	200x200	24-bit	color(RGB)
baby_bear.raw	200x200	24-bit	color(RGB)

Problem 2: Digital Half-toning

mandrill.raw	512x512	8-bit	gray
sailboat.raw	512x512	24-bit	color(RGB)

Problem 3: Morphological Processing

Horseshoe.raw	91x108	8-bit	gray
Horse1.raw	360x275	8-bit	gray
Horse2.raw	390x196	8-bit	gray
Camel.raw	352x323	8-bit	gray
Deer.raw	300x366	8-bit	gray

References

[1] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," SPIE MILESTONE SERIES MS, vol. 154, pp. 139–143, 1999.

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

[3] http://www.dabi.temple.edu/~shape/MPEG7/dataset.html

[4] http://images.google.com/

[5] http://sipi.usc.edu/database/