Problem 1:

(a) Neighbor padding: 12, 12, 13, 12, 12, 13, 3, 3, 1 (0.5 pts.)
Mean: 9 (0.5 pts.)
Median: 12 (0.5 pts.)
Zero padding: 0, 0, 0, 0, 12, 13, 0, 3, 1 (0.5 pts.)
Mean: 3.22 (0.5 pts.)
Median: 0 (0.5 pts.)

Histogram:

Pixel values	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pdf*25	2	1	4	1	0	1	1	0	1	2	1	2	3	5	1	0
Cdf*25	2	3	7	8	8	9	10	10	11	13	14	16	19	24	25	25

(2 pts.)

Therefore, the value for original image center should be 24/25 = 0.96. Since the maximum value of 4-bit grayscale image is 15, so 15*0.96 = 14.4, select the integer part 14. (2 pts.)

(b) False. LPF might reduce noise, but certainly will not sharpen the image as it will blur the edges and other high frequency content of the image. (3 pts.)
True. Gaussian filtering operation is linear and hence can be broken down into row and column operations. Full credit was given even if the following equation was not provided. (3 pts.)

$$G(x,y) = k\ \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) = \sqrt{k}\ \exp\left(-\frac{x^2}{2\sigma^2}\right)\sqrt{k}\ \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

False. Since Sobel filter will contain more edge information, i.e. (I*F), the method noise will have less edge information (as compared to Gaussian filter). (3 pts.)

(c) He applied the mean filter before median filter, which spreads the impulsive noise to its neighborhood, thus results larger dark spot in the denoised image. (3 pts.)
(d)The Gaussian filter since it is better at preserving high frequency component (edge) of images. (3 pts.)

Problem 2:
(a)
Output matries:
Apply Gx: (1 pts.)

0	40	40	0
0	20	20	0
0	-20	-20	0
0	-40	-40	0

Apply Gy: (1 pts.)

0	0	0	0
40	20	-20	-40
40	20	-20	-40
0	0	0	0

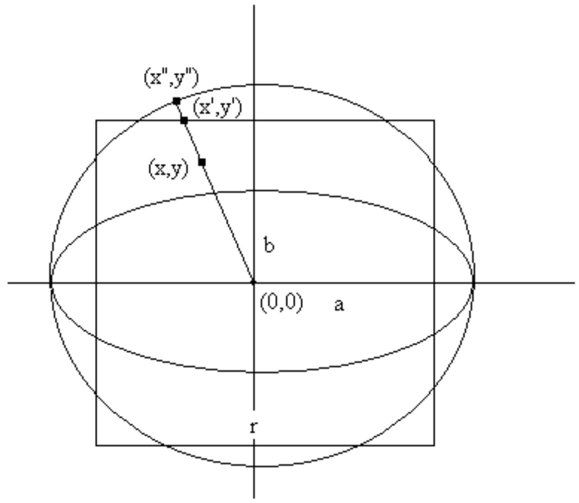
Gx: vertical edge detector; (1 pts.) Gy: horizontal edge detector. (1 pts.)
(b) Zero-crossing (2 pts.)
(c) b (1 pts.)
Notice that we did not ask under what scenario LoG would work best. "Less noise" is good for both LoG and Sobel; but if it is the case, Gaussian filter will smooth the image, potentially blurring some edges. On the other hand, Sobel also has some smoothing effect, which is not as strong as the Gaussian filter but might be enough to handle a low level of noise. Therefore, Sobel works better than LoG under "less noise" scenario in general.
Note that Sobel operator is also susceptible to noise (although not as much as the Laplacian operator). Therefore, if we have a high level of noise, Sobel will not perform well. The use of Gaussian filter in LoG, however, might be advantageous because of its denoising effect. (1 pts.)

LoG works better for weaker edges because zero-crossing is not so dependent on the selection of threshold as in Sobel. (1 pts.)
Grading criteria:
- Selection itself is only worth 1 point
- Partially correct selection (weaker edges OR more noise) can get partial credit if proper justification is provided. Partial credit might be 1 or 2 points depending on the justifications.

(d) High pass filter (2 pts.) Smaller (1 pts.) reason (2 pts.)

Problem 3

Assume the size of the original image is r*r and the output image a*b.



+2

First, we warp the square image into a **disk** image with radius a: For a boundary pixel (x', y') of original image, we extend the line connecting itself and origin (0, 0) to reach the boundary of disk image. Define the scaling factor on this line: +2

$$s_{line} = \frac{a}{\sqrt{(x')^2 + (y')^2}}$$

All the pixels (x, y) in this line will scale along the line with this scaling factor. Thus, we can get a disk image. Next, we warp the disk image into the output (**elliptical**) image. +2

From the above figure, this can be easily done by "compress" (scaling) along y-axis but keep the x-axis value unchanged. +2

(2) (5 points) You cannot (+4) warp a square image into an elliptical image using 9 control points. If you use the polynomial method, then you need to ensure that each point on the boundary of square image be mapped to the boundary point of elliptical image. That is impossible. +2

You may receive 3-4 points for (a) depends on the correctness of the mapping method used.

Problem 4

(a.1)
L3L3: brightness

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \quad * \ 1/36$$

L3E3: image has obvious vertical edge

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \quad * \ 1/12$$

S3S3: spot

$$\begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix} \quad * \ 1/4$$

(a.2)

LL	LS	LE	SL	SS	SE	EL	ES	EE
1.0000	0.2263	0.2856	0.0570	0.0689	0.0624	0.0862	0.0993	0.0970
1.0000	0.0713	0.0744	0.1216	0.0794	0.0773	0.1512	0.0713	0.0695
1.0000	0.1781	0.1575	0.1006	0.2143	0.1520	0.1262	0.2191	0.1698
1.0000	0.2058	0.2123	0.1415	0.2057	0.1943	0.1517	0.2246	0.2155

Energy Responses

A [2pt]: LE/LS

B [2pt]: EL/SL

C [2pt]: SS/ ES

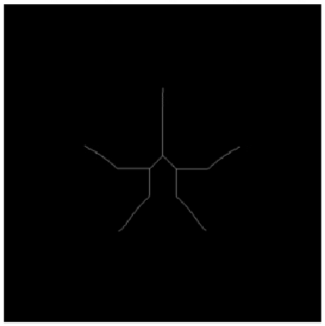
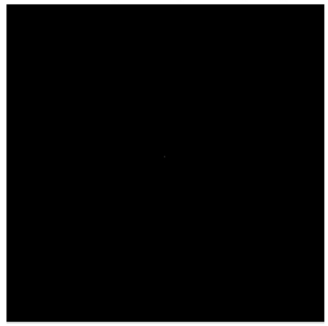
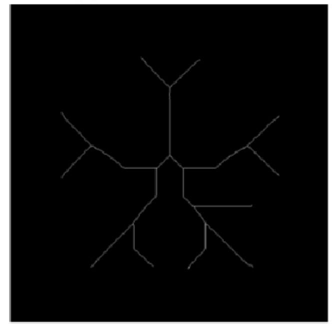
D [2pt]: ES/EE

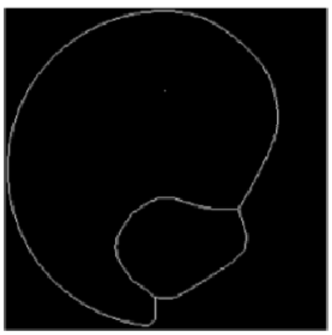
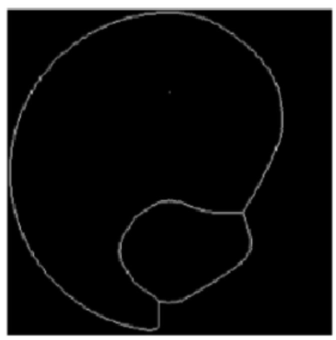
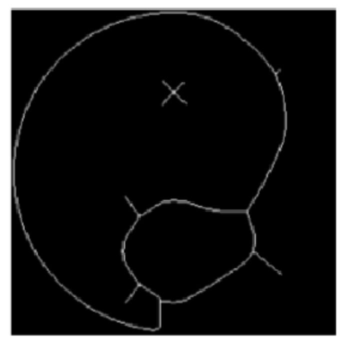
(b)
1) Preprocessing: Subtract the local mean from each pixel [1pt]. (Eliminate the effect of the different illumination within the same texture aera. [1pt])

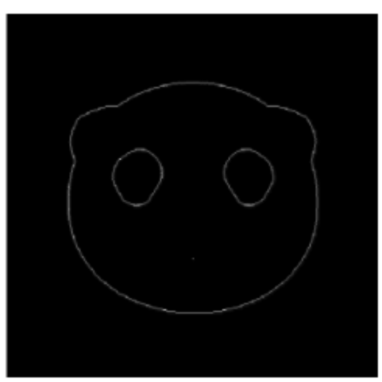
2) Laws feature extraction: Apply 5x5 Laws filters to the input image and get 25 gray–scale images. [1pt]
3) Local energy feature computation: Use a window approach to computer the energy measure for each pixel based on the results from Step2. In other words, for each feature dimension, to compute the square sum of all the neighbor pixels' values, and then normalize the square sum by the size of window (NxN). N is typically much larger than 5, so this windowing procedure is referred as "macro-window energy computation". After this step, we can obtain 25-D energy feature vector for each pixel. [1pt] (extract powerful features representing different texture[1pt])
4) Energy feature normalization: the range of all features should be normalized to the range of [0,1]. [1pt] (each feature contributes approximately proportionately when computing Euclidean distance.[1pt])
5)Segmentation: Use the k–means algorithm to perform segmentation on the composite texture images. (Classify each pixels to achieve segmentation) [1pt]

Problem 5

Thinning: Shrinking: Skeletonizing:

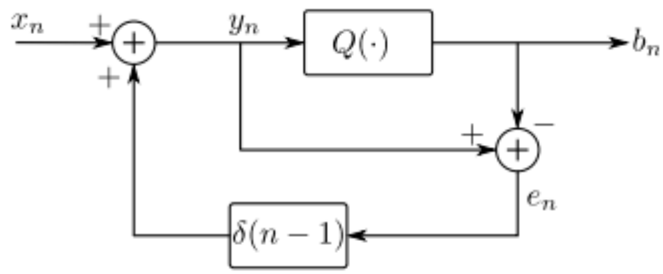





Problem 6

6.1. You get 3 points for the flow-diagram



6.2 You would get 4 points if bn are correct.

n	1	2	3	4	5	6	7	8	9	10
x_n	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
y_n	0.25	0.5	0.75	0	0.25	0.5	0.75	0	0.25	0.5
b_n	0	0	1	0	0	0	1	0	0	0
e_n	0.25	0.5	−0.25	0	0.25	0.5	−0.25	0	0.25	0.5

6.3 You would get 2 points for dn, 1 points for \sum_{dn}

$$d_n = x_n - b_n = x_n - (y_n - e_n) = x_n - (x_n + e_{n-1} - e_n) = e_n - e_{n-1}$$

$$\sum_{n=1}^{N} d_n = \sum_{n=1}^{N} (e_n - e_{n-1}) = e_N - e_0 = e_N$$

6.4 You would get 2 points if you figured out one of them. 3 points for both.

1) It tells us that the accumulated error is bounded. Now we prove that e_N is bounded.

   i) When $N = 0$, $e_0 = 0$ is bounded.

   ii) We assume that e_k is bounded, then $y_{k+1} = x_{k+1} + e_k$ is bounded since x_{k+1} is bounded as well. Then $e_{k+1} = y_{k+1} - b_{k+1}$ is bounded since y_{k+1} is bounded and $b_{k+1} = Q(y_{k+1})$ is bounded. Thus e_k is bounded and e_{k+1} is bounded.

2) It tells us that the total average of the signal is approximately maintained.

$$\sum_{n=1}^{N} d_n = \sum_{n=1}^{N} (x_n - b_n) = e_N$$
$$\Rightarrow \sum_{n=1}^{N} b_n = \sum_{n=1}^{N} x_n - e_N$$
$$\Rightarrow \frac{1}{N}\sum_{n=1}^{N} b_n = \frac{1}{N}\sum_{n=1}^{N} x_n - \frac{1}{N} e_N$$

So the local average of the output is approximately the same as the local average of the input.

6.2 get 2 points for T4

$$T_4 = \begin{bmatrix} 87.65 & 151.4 & 103.6 & 167.3 \\ 215.2 & 23.91 & 231.1 & 39.84 \\ 119.5 & 183.3 & 71.72 & 135.5 \\ 247.1 & 55.78 & 199.2 & 7.98 \end{bmatrix}$$

2 points for the half-toning results

$$\begin{matrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{matrix}$$

1 point for counting

125 region = 8, 25 region = 2, 75 region = 5, 225 region = 14

1 point for the explanation.

The number of “1” value pixels in each region implies that overall gray scale value (brightness or darkness) of certain region.