

Image Processing

Contents

1	Exercise 1	2
2	Exercise 2	2
3	Exercise 3	2
4	Exercise 4	2
5	Exercise 5	2
6	Exercise 6	2
7	Exercise 7	2
8	Exercise 8	3
9	Graphs	4
9.1	Exercise 1	4
9.2	Exercise 2	5
9.3	Exercise 3	6
9.4	Exercise 4	7
9.5	Exercise 5	8
9.6	Exercise 6	10
9.7	Exercise 7	12
9.8	Exercise 8	15
10	Codes	18
10.1	Exercise 1	18
10.2	Exercise 2	18
10.3	Exercise 3	19
10.4	Exercise 4	19
10.5	Exercise 5	21
10.6	Exercise 6	22
10.7	Exercise 7	23
10.8	Exercise 8	24

1 Exercise 1

The images show 3 circles with different colors (red, green, blue) close to the center. As the circles expand, the color fades away. The greater the p value the faster the color fades away.

2 Exercise 2

The images generated show randomly colored dots. When the code is recompiled. The dots appear as if they were scattered in a different order.

3 Exercise 3

The 5 point moving average results in a smoother plot than the 2 point moving average. This is due to the 5 point moving average averaging larger samples of data. The 5 point moving average also has a lower peak than the 2 point moving average

4 Exercise 4

The derivative approximation using 11 points has a lower peak and a lower minimum than the derivative approximation using 51 points. Furthermore, the separation between points is higher using 11 points than when using 51 points. Lastly, the approximations using 51 points is more intricate and has more curves than when using 11 points.

5 Exercise 5

The 10x10 blur applies a scaled 10 row, 10 column blur to the coins image. The resulting image appears to be blurred equally horizontally and vertically. The 2x50 blur applies a a scaled 2 row, 50 column blur to the coins image. The resulting image appears to have a wider blur horizontally than vertically. The 50x2 blur applies a a scaled 50 row, 2 column blur to the coins image. The resulting image appears to have a wider blur vertically than horizontally.

6 Exercise 6

According to the MATLAB documentation of the `conv2` function, the `valid` option returns the convolution parts computed with unpadded edges, whereas `conv2` with the same option returns the central part of the convolution such that the returned element has the same size as the first parameter fed to `conv2`. The resulting images using the `valid` option are slightly cropped and smaller when compared to the images using the same option.

7 Exercise 7

1. The first image shows the result of a 5x5 Gaussian Blur. The Gaussian blur resulting image is less blurry than the 10x10 blur.
2. The second image shows the vertical edges of the coins using a Prewitt operator that detects changes in gray level from left to right.
3. The third image shows the horizontal edges of the coins using a Prewitt operator that detects changes in gray level from top to bottom.
4. The fourth image displays the normalized square root of the sum of squares of the previous 2 Prewitt operators. The resulting image shows the edges of the coins in bold white, the background in black, and the edges of drawings on the coin in faint white.
5. The fifth image displays a 3x3 third edge detection kernel. The resulting image displays the edges of the coins in white, the background in dark gray, and the edges of drawings on the coin in faint black.

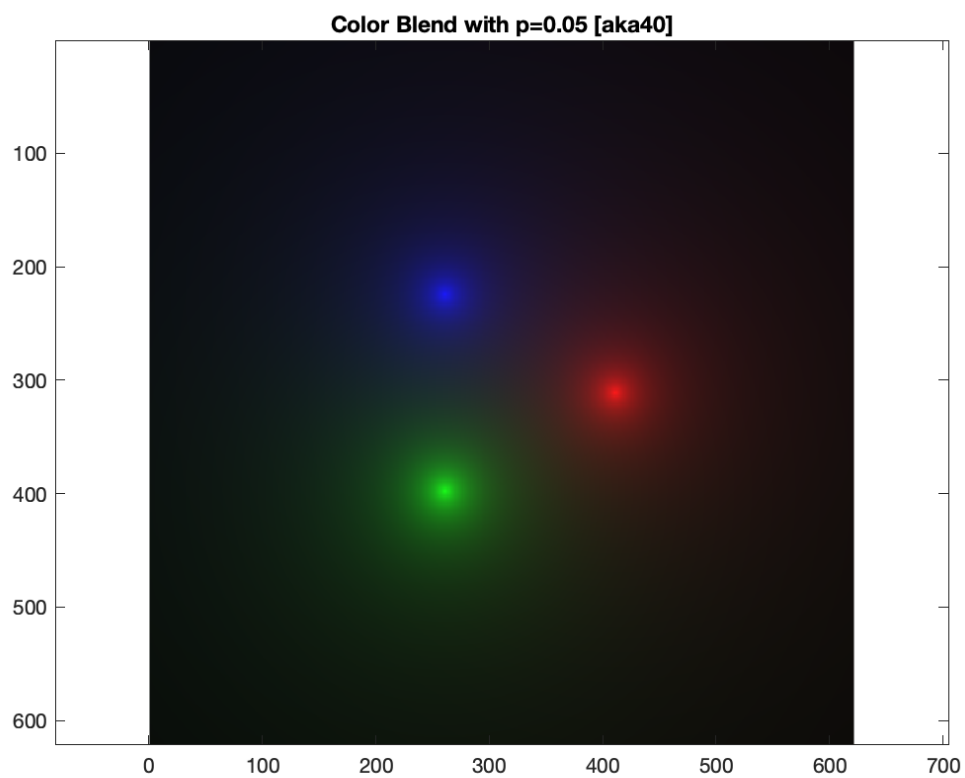
The fourth image displays better edge detection than the fifth due to higher contrast in the colors of the edges and the background. The edges are also more clearly outlined in the fourth image than in the fifth.

8 Exercise 8

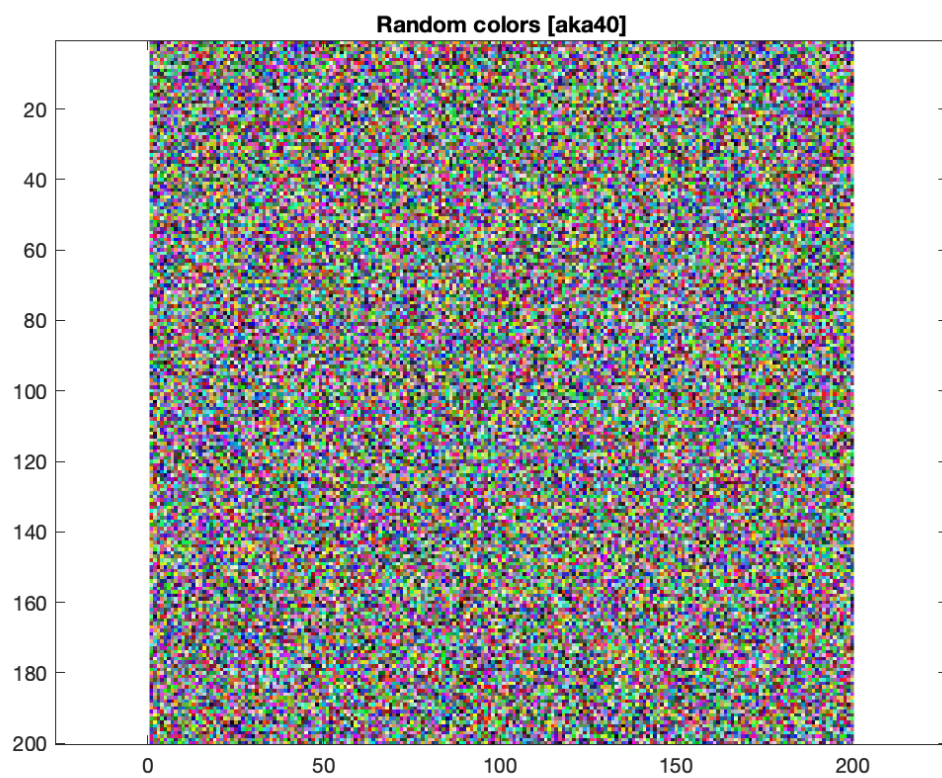
1. The first image shows the original image with no blurs.
2. The second image shows the result of applying a 21x21 box blur to the original image. This blur looks a standard intense blur.
3. The third image shows the result of applying a 5x5 Gaussian Blur. The Gaussian blur resulting image is very subtle and less blurry than the 21x21 box blur.
4. The fourth image shows the result of applying a 3x3 Sobel operator blur that detects change in color horizontally (going from left to right). The resulting image shows colored vertical edges of the original image.
5. The fifth image shows the result of applying a 3x3 Sobel operator blur that detects change in color vertically (going from top to bottom). The resulting image shows colored horizontal edges of the original image.
6. The last image displays the square root of the sum of squares of the previous 2 Sobel operators. The resulting image displays the colored horizontal and vertical edges in the original image.

9 Graphs

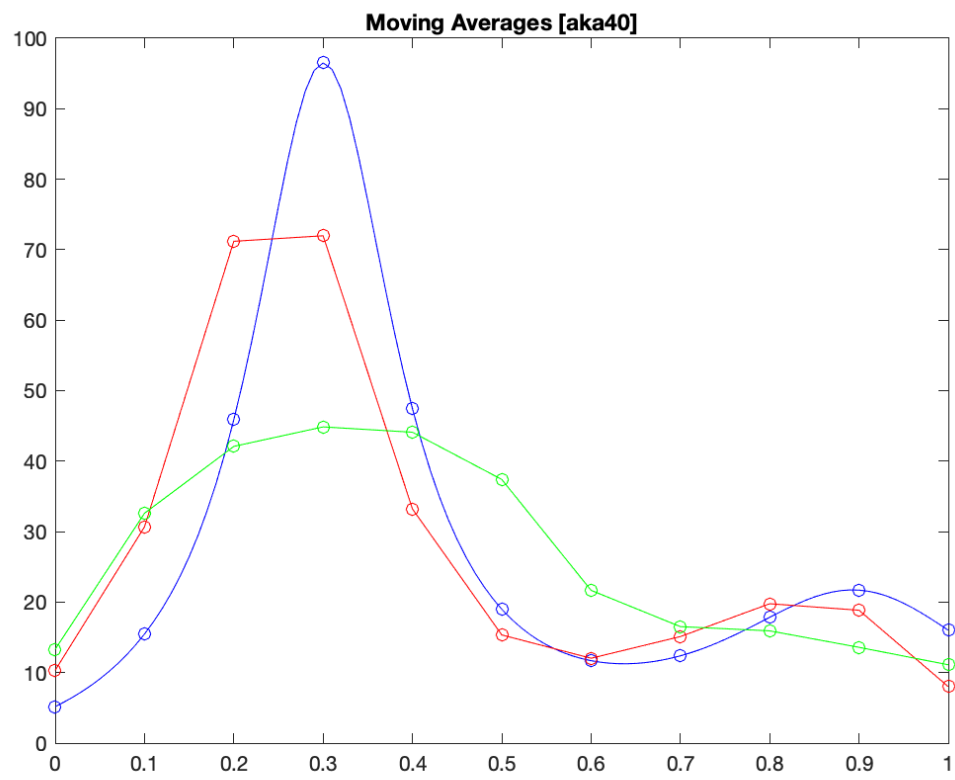
9.1 Exercise 1



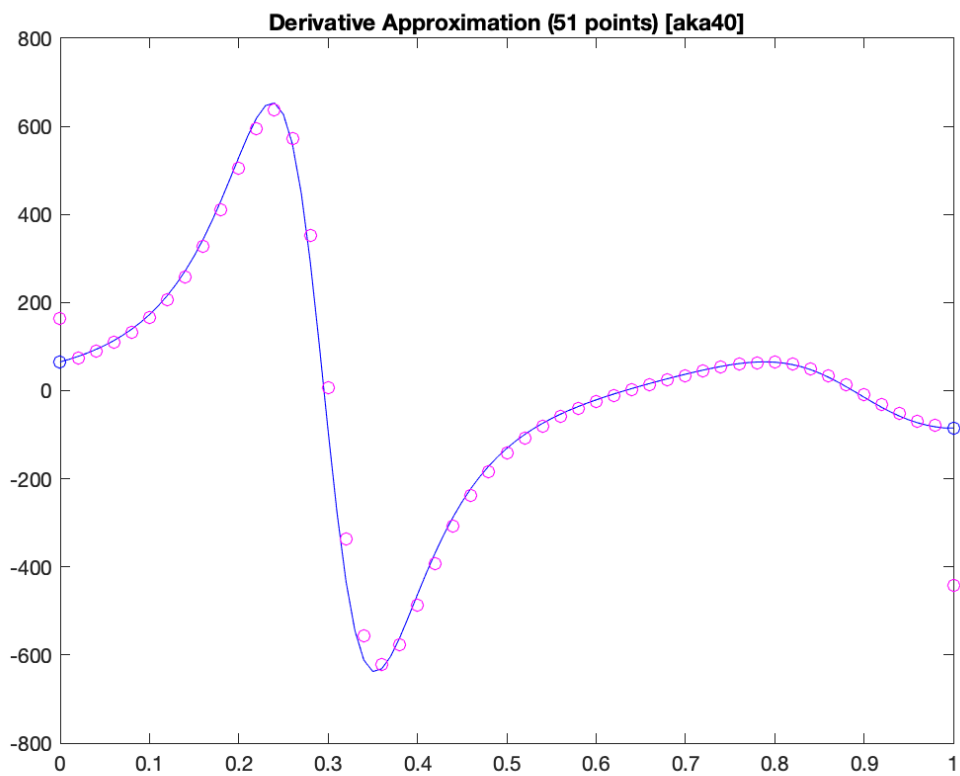
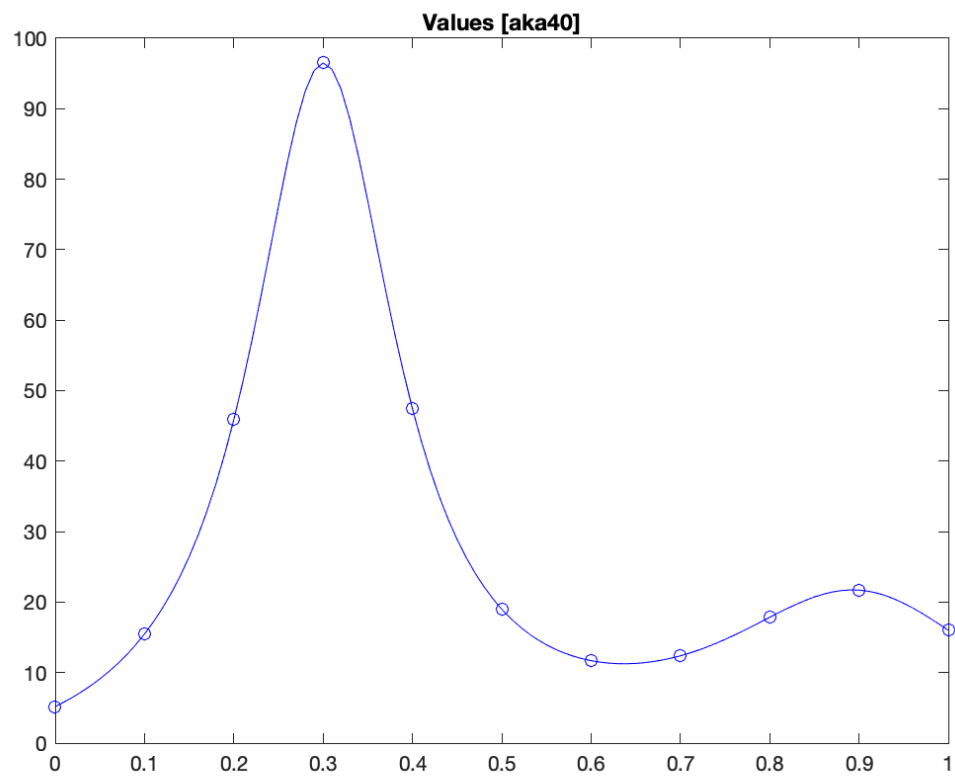
9.2 Exercise 2



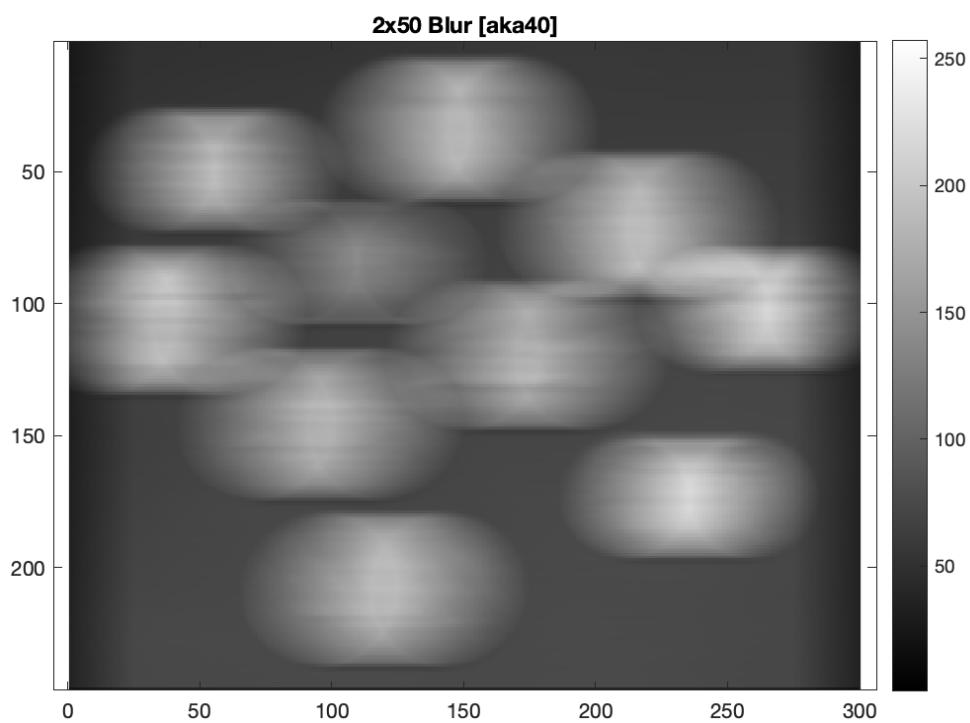
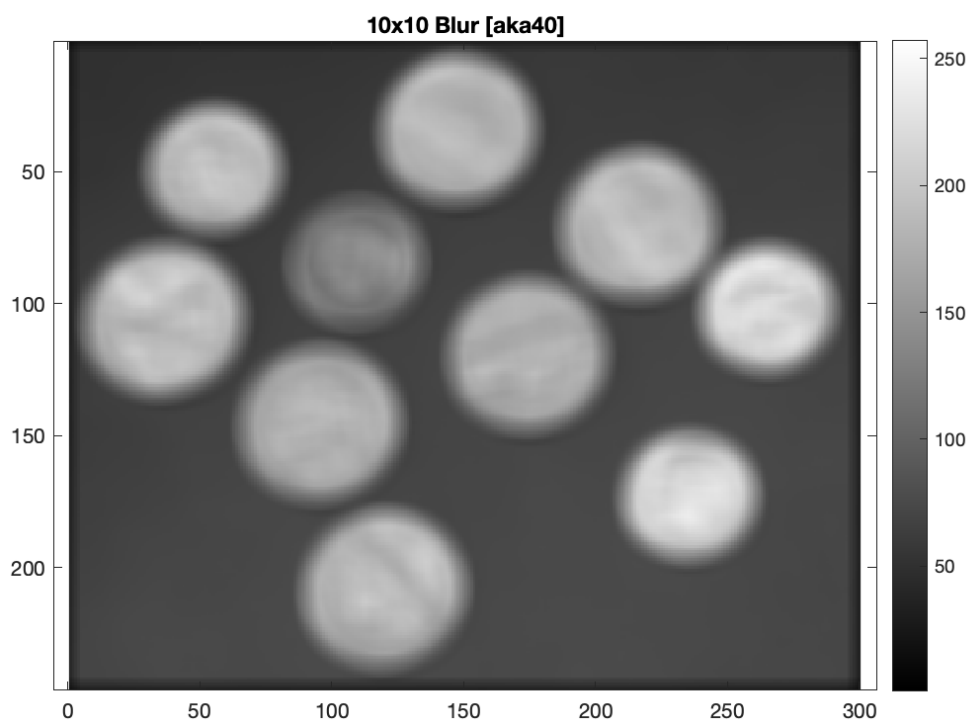
9.3 Exercise 3

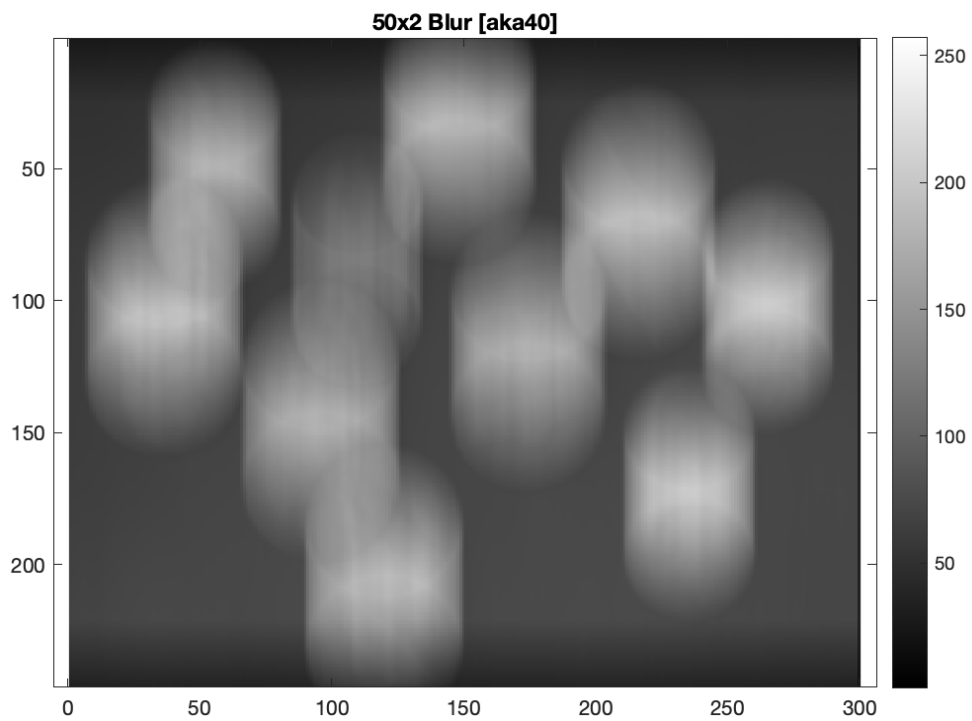


9.4 Exercise 4

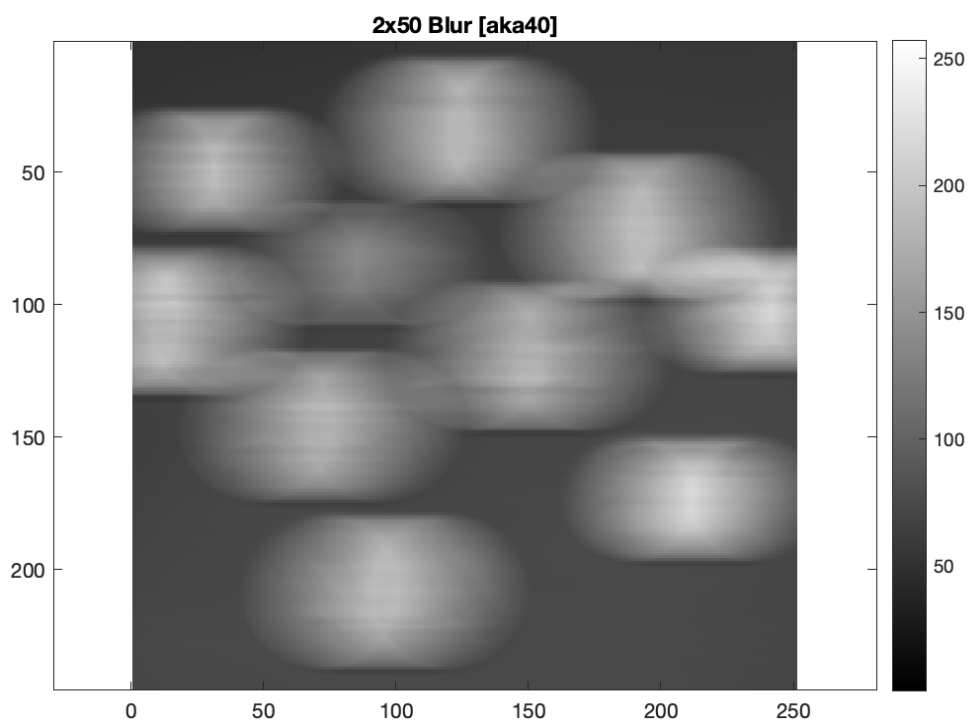
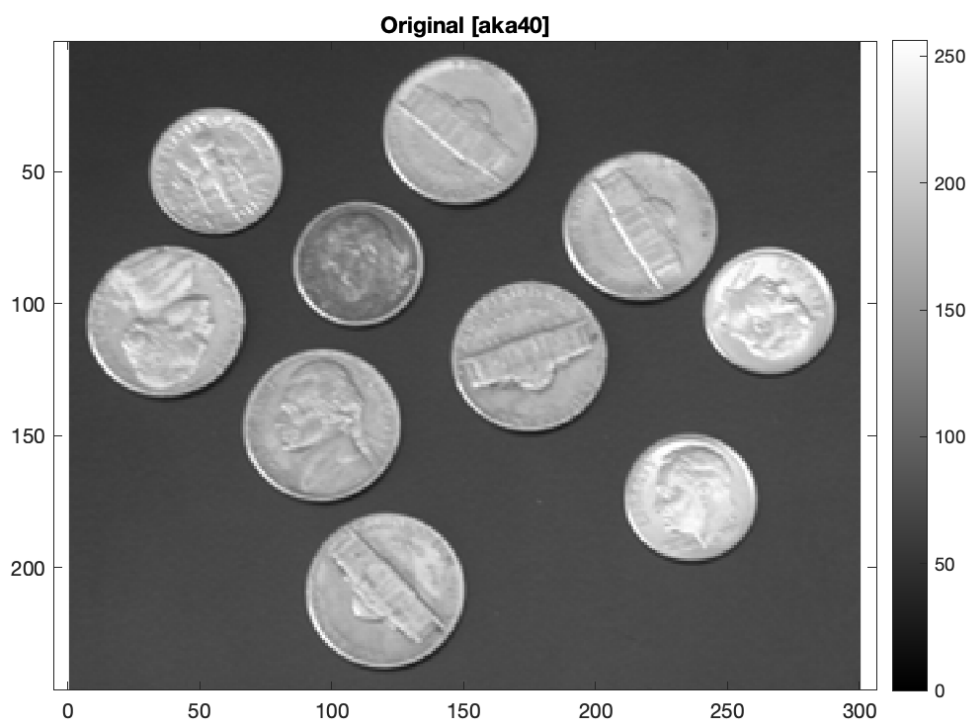


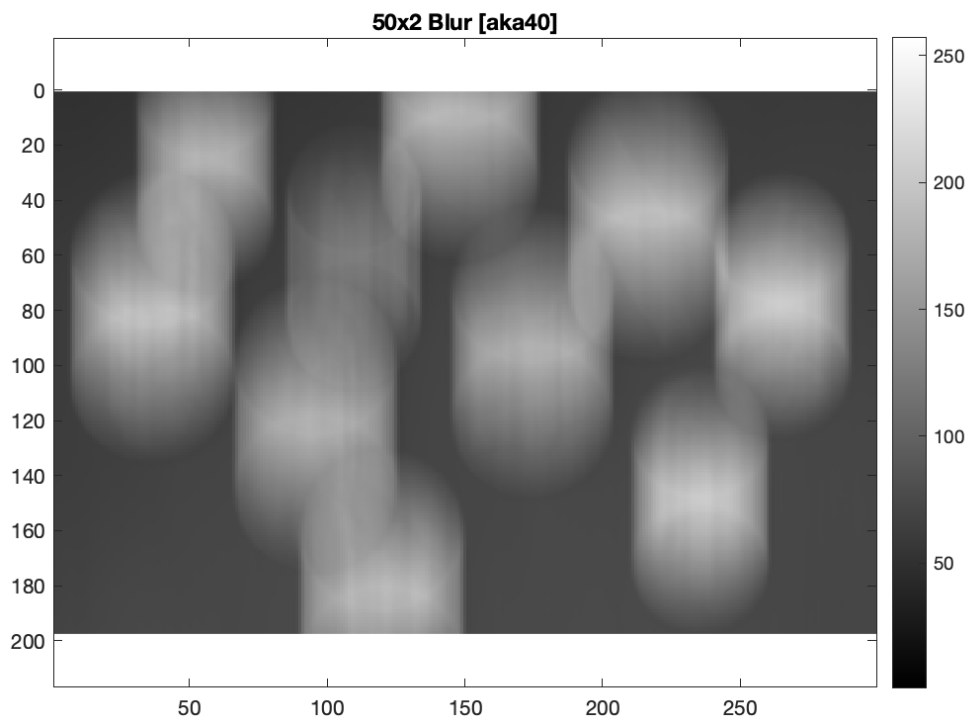
9.5 Exercise 5



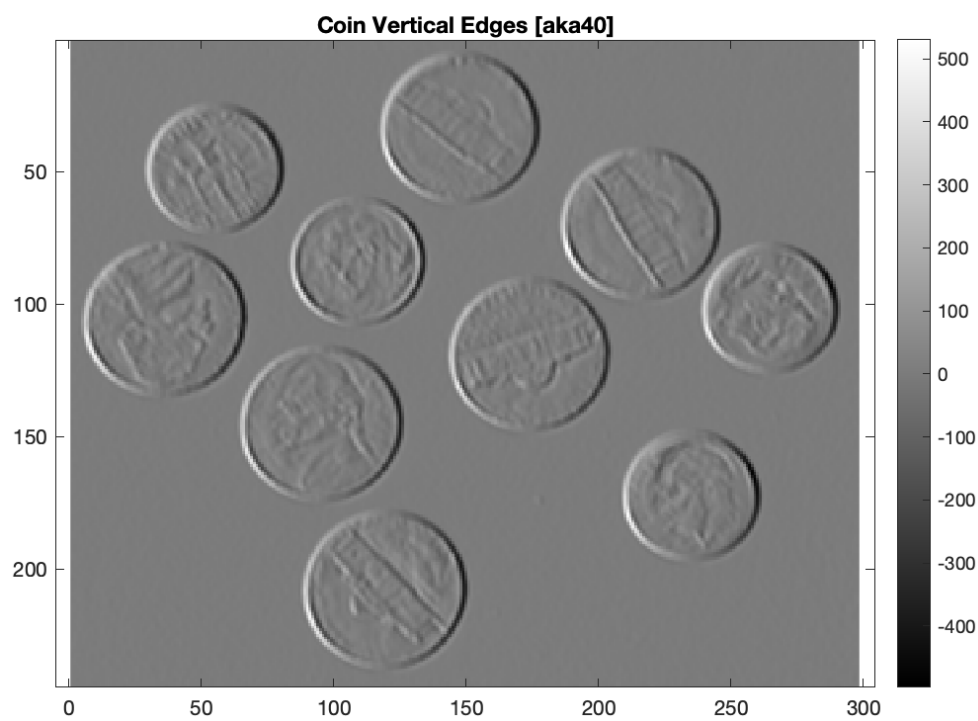
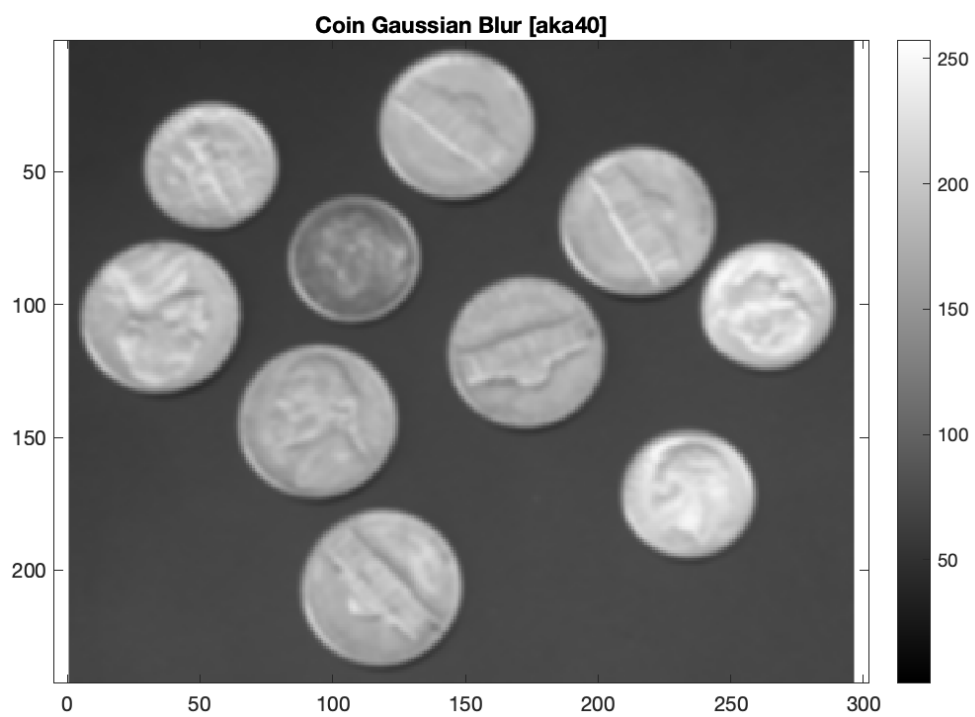


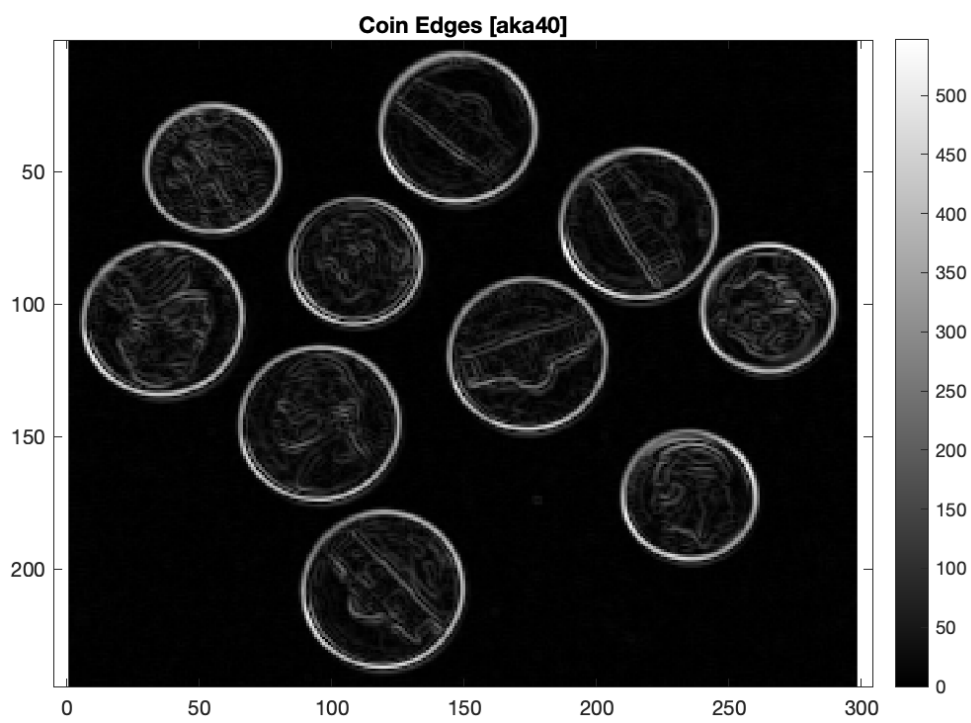
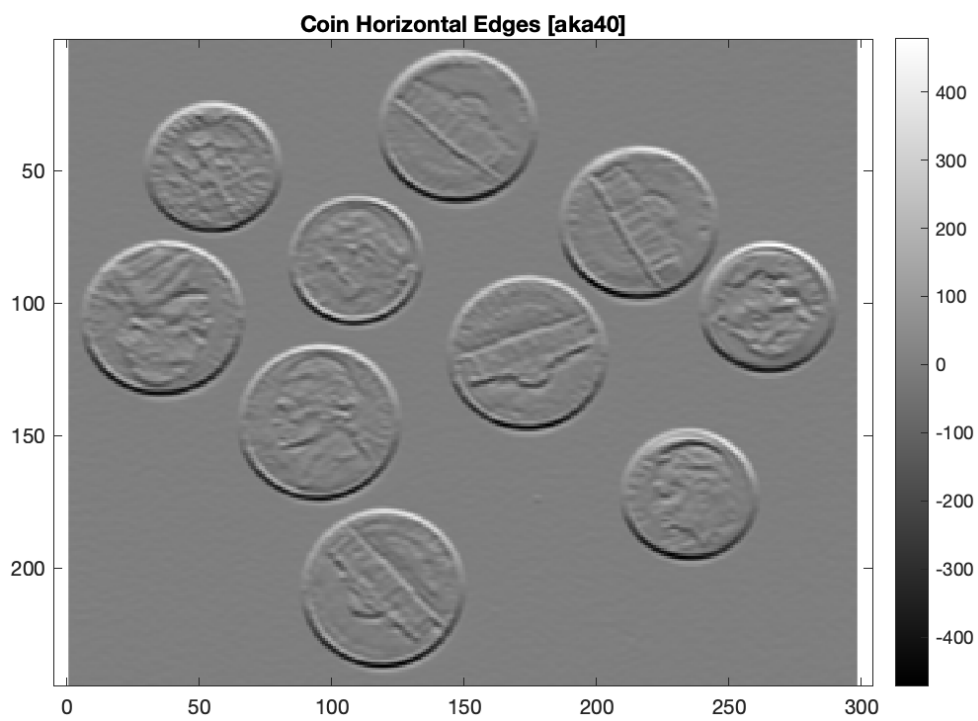
9.6 Exercise 6

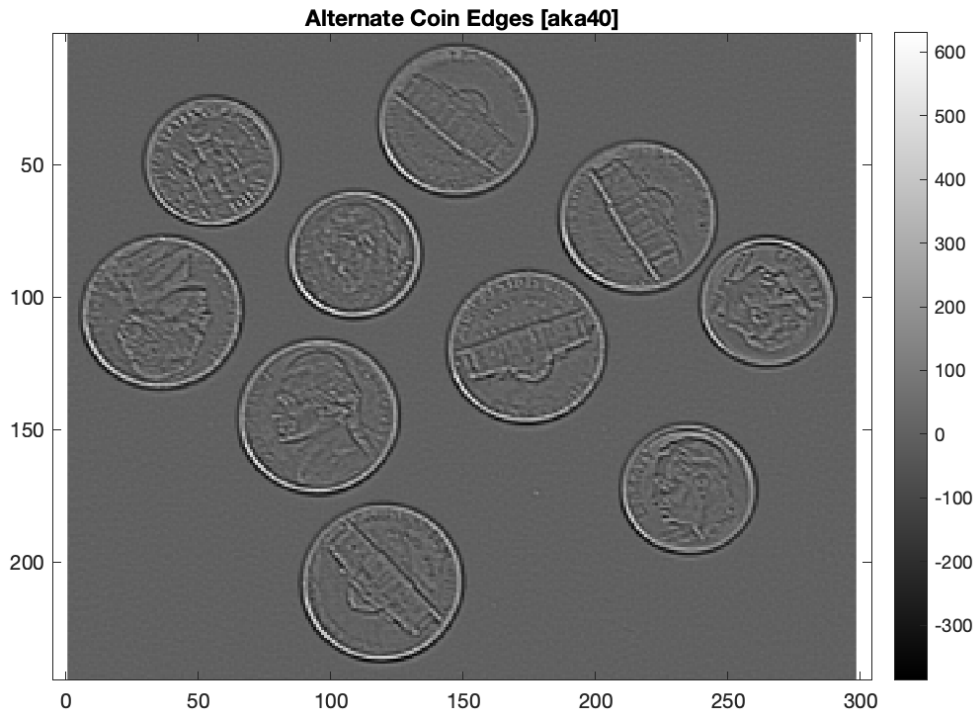




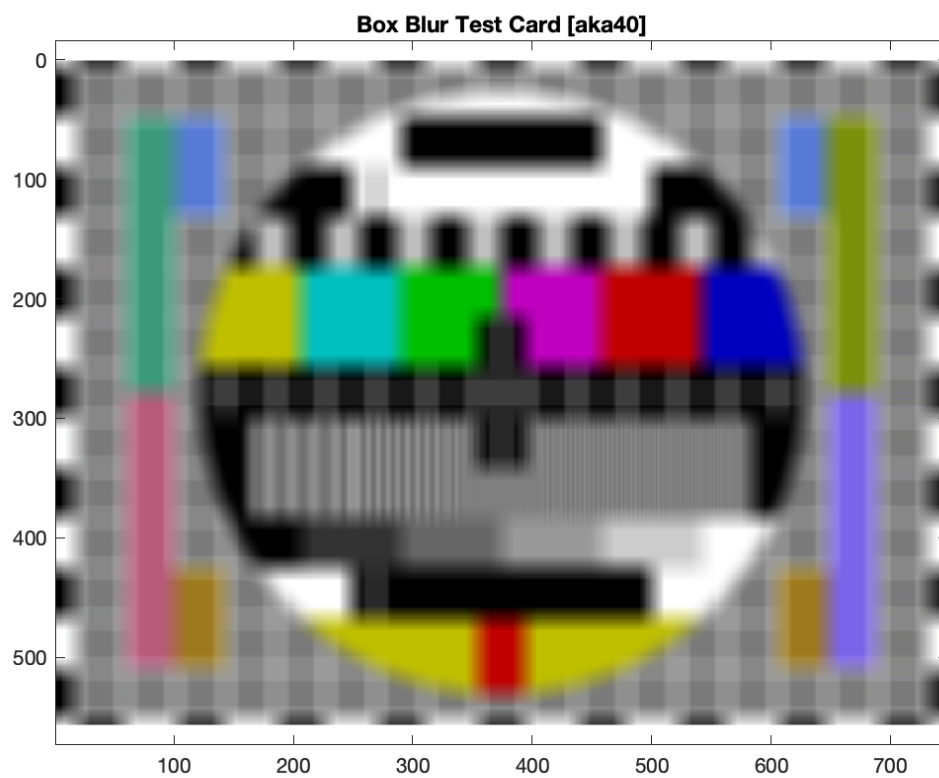
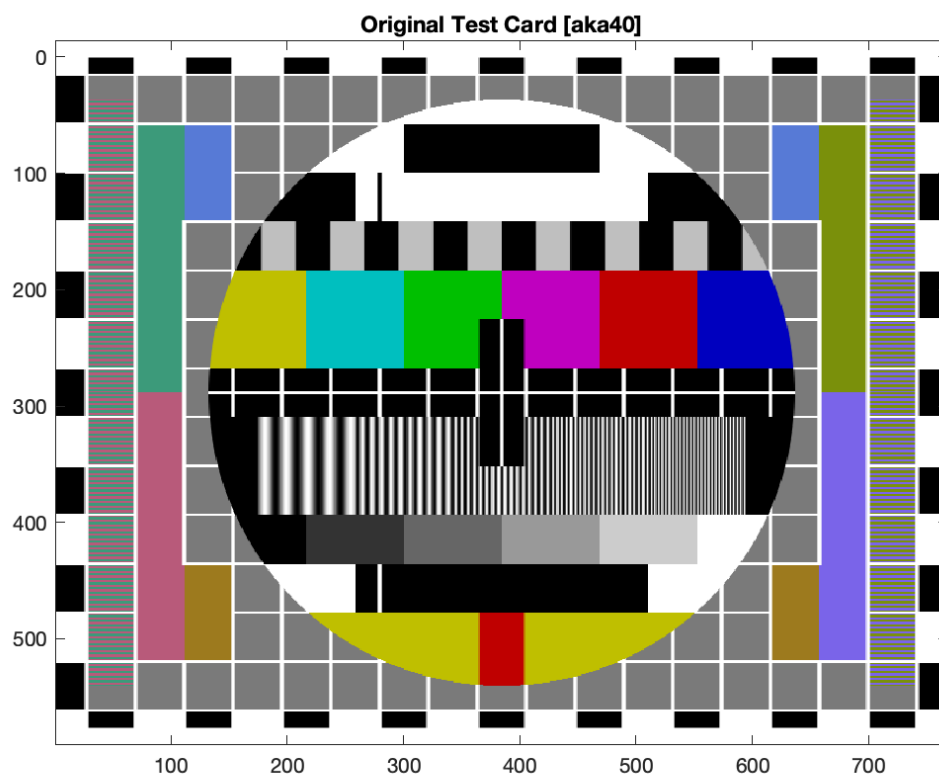
9.7 Exercise 7

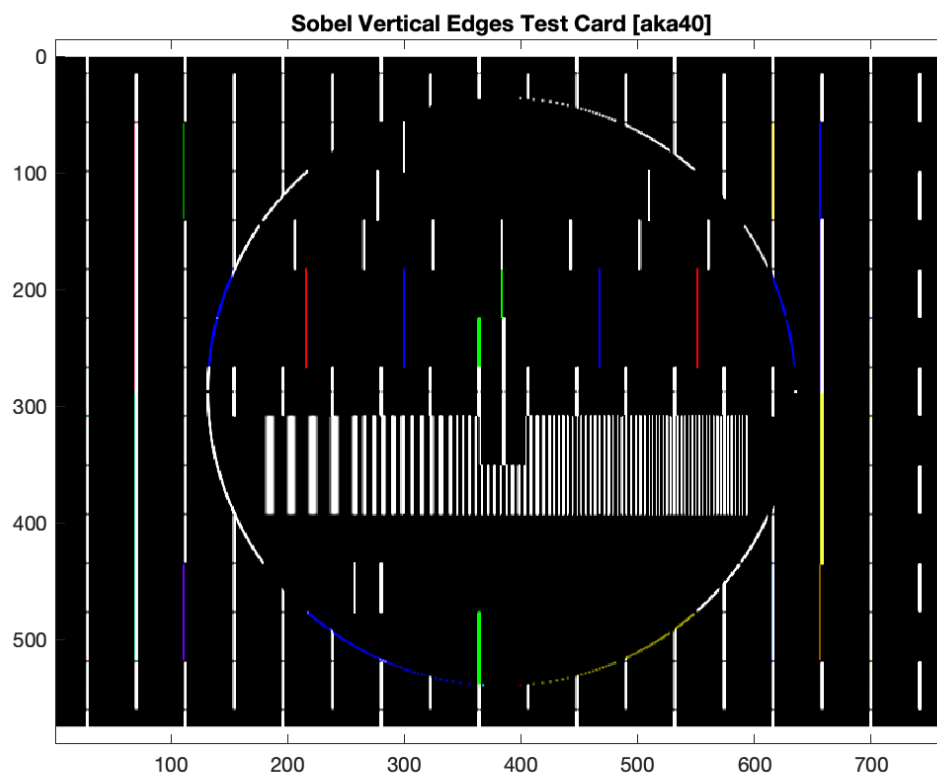
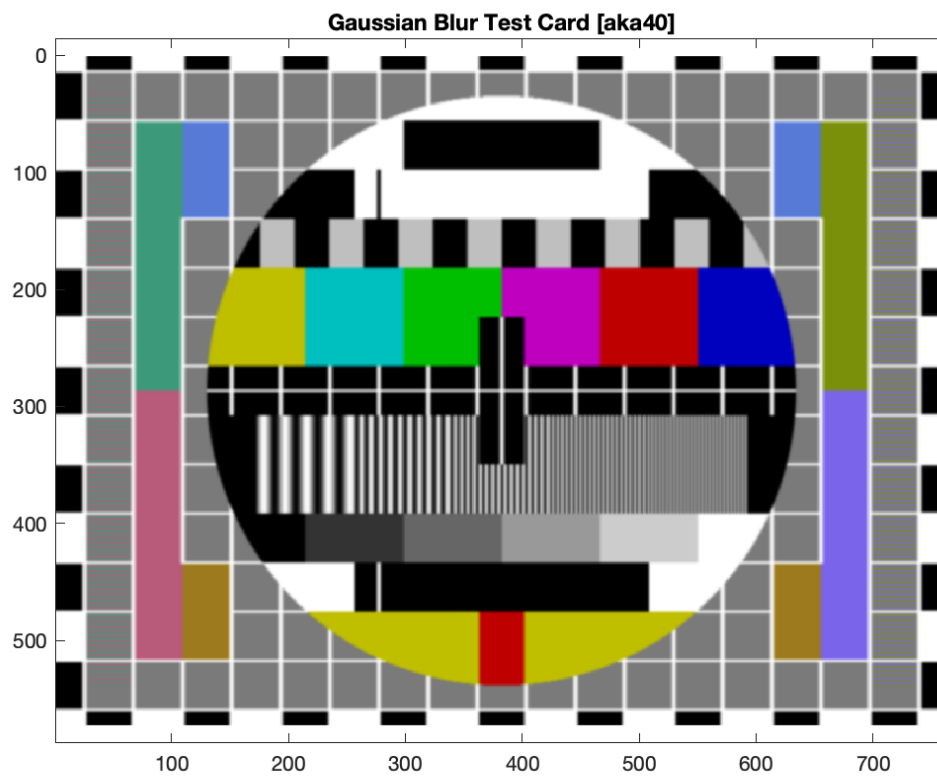


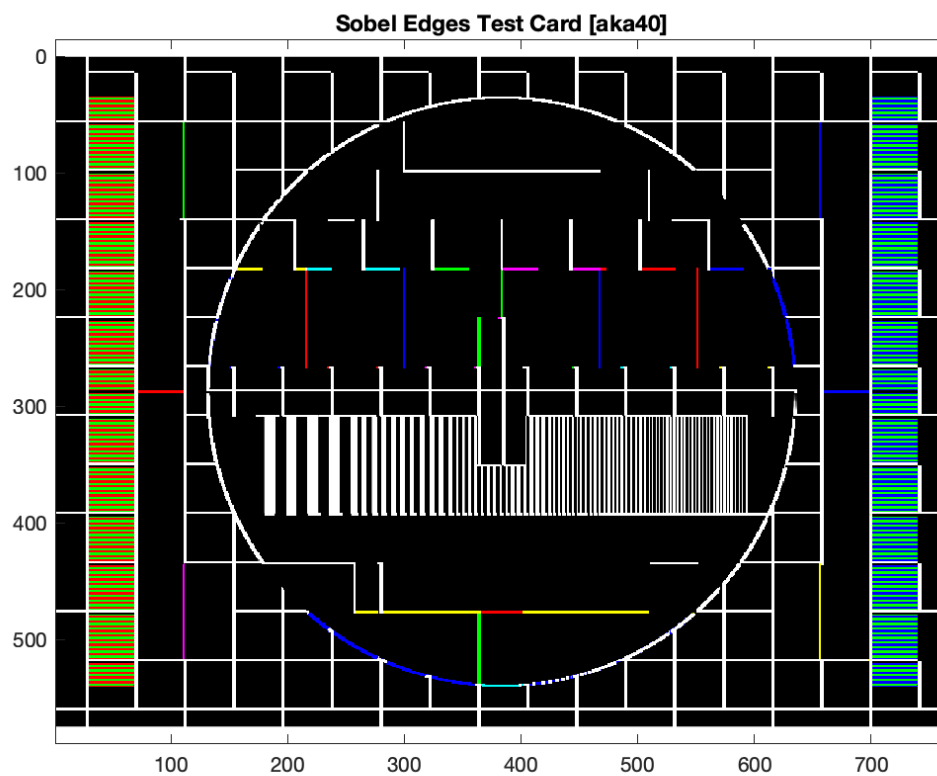
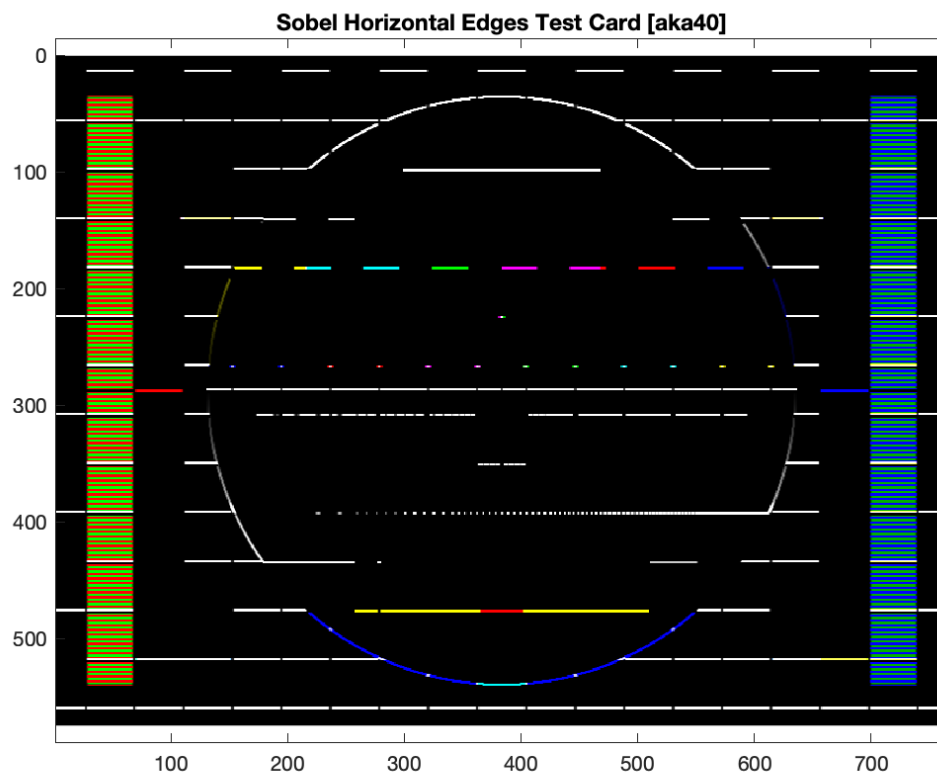




9.8 Exercise 8







10 Codes

10.1 Exercise 1

```
1 clear
2 rad = 100;
3 del = 10;
4 [x, y] = meshgrid((-3*rad-del):(3*rad+del));
5 [rows, cols] = size(x);
6 p=0.05;
7 dist = @(x, y, xc, yc) sqrt((x-xc).^2+(y-yc).^2);
8 venn_img = zeros(rows, cols, 3);
9 venn_img(:,:,1) = 1./(1+p*sqrt((x-rad.*cos(0)).^2+(y-rad.*sin(0)).^2));
10 venn_img(:,:,2) = 1./(1+p*sqrt((x-rad.*cos(2*pi/3)).^2+(y-rad.*sin(2*pi/3)).^2));
11 venn_img(:,:,3) = 1./(1+p*sqrt((x-rad.*cos(4*pi/3)).^2+(y-rad.*sin(4*pi/3)).^2));
12 figure(1); clf
13 image(venn_img)
14 axis equal
15 title("Color Blend with p=0.05 [aka40]")
16
17 print -dpng IP1_EX1_Plot1.png
18 p=0.005;
19 dist = @(x, y, xc, yc) sqrt((x-xc).^2+(y-yc).^2);
20 venn_img = zeros(rows, cols, 3);
21 venn_img(:,:,1) = 1./(1+p*sqrt((x-rad.*cos(0)).^2+(y-rad.*sin(0)).^2));
22 venn_img(:,:,2) = 1./(1+p*sqrt((x-rad.*cos(2*pi/3)).^2+(y-rad.*sin(2*pi/3)).^2));
23 venn_img(:,:,3) = 1./(1+p*sqrt((x-rad.*cos(4*pi/3)).^2+(y-rad.*sin(4*pi/3)).^2));
24 figure(2); clf
25 image(venn_img)
26 axis equal
27 title("Color Blend with p=0.005 [aka40]")
28
29 print -dpng IP1_EX1_Plot2.png
30
31 % The images show 3 circles with different colors at the center (red, green, blue). As th
32 % circles expand, the color fades away. The greater the p value the faster
33 % the color fades away.
34 %
```

10.2 Exercise 2

```
1 clear
2 array = zeros(200,200,3);
3 array(:,:,1)= rand(200,200);
4 array(:,:,2)= rand(200,200);
5 array(:,:,3)= rand(200,200);
6 figure(1);clf
7 image(array)
8 axis equal
9 title("Random colors [aka40]")
10
11 print -dpng IP1_EX2_Plot1.png
12
13 % The images randomly colored dots. When the code is recompiled. The dots
14 % appear as if they were scattered in a different order.
```

10.3 Exercise 3

```
1 clear
2 tc = linspace(0,1, 101);
3 xc = humps(tc);
4 td = linspace(0,1,11);
5 xd = humps(td);
6 figure(1),clf
7 plot (tc, xc, 'b-')
8 hold on
9 plot (td, xd, 'bo')
10
11 movingAverage2 = conv(xd, ones(1,2)/2, 'same')
12 plot (td, movingAverage2, 'ro-')
13
14 movingAverage5 = conv(xd, ones(1,5)/5, 'same')
15 plot (td, movingAverage5, 'go-')
16 title("Moving Averages [aka40]")
17 hold off
18
19 print -dpng IP1_EX3_Plot1.png
20
21 % The 5 point moving average results in a smoother plot than the 2 point
22 % moving average. This is due to the 5 point moving average average larger
23 % samples of data this is. The 5 point moving average also has a lower peak than
24 % the 2 point moving average
```

10.4 Exercise 4

```
1 clear
2 tc = linspace(0,1, 101);
3 xc = humps(tc);
4 deltatc = tc(2) - tc(1)
5 td = linspace(0,1,11);
6 xd = humps(td);
7 deltatd = td(2) - td(1)
8
9 figure(1),clf
10 plot (tc, xc, 'b-')
11 hold on
12 plot (td, xd, 'bo')
13 hold off
14
15 title("Values [aka40]")
16
17 figure(2); clf
18 twopointdiff = diff(xc)/deltatc;
19 twopointdiff(end+1) = twopointdiff(end);
20 plot(tc,twopointdiff, 'b-');
21
22 hold on
23 plot(tc(1:10:end), twopointdiff(1:10:end), 'bo');
24
25 con1 = conv(xd, [1,0,-1]/2/deltatd, 'same')
26 plot (td, con1, 'mo')
27 title("Derivative Approximation (11 points) [aka40]")
28 hold off
29 print -dpng IP1_EX4_Plot11.png
```

```

30
31 figure(3); clf
32 tc = linspace(0,1, 101);
33 xc = humps(tc);
34 deltadc = tc(2) - tc(1)
35 td = linspace(0,1,51);
36 xd = humps(td);
37 deltatd = td(2) - td(1)
38 twopointdiff = diff(xc)/deltadc;
39 twopointdiff(end+1) = twopointdiff(end);
40 plot(tc,twopointdiff, 'b-');
41
42 hold on
43 plot(tc(1:100:end), twopointdiff(1:100:end), 'bo');
44
45 con1 = conv(xd, [1,0,-1]/2/deltatd, 'same')
46 plot (td, con1, 'mo')
47 title("Derivative Approximation (51 points) [aka40]")
48
49 %
50
51 print -dpng IP1_EX4_Plot51.png

```

10.5 Exercise 5

```
1 clear
2 x = imread('coins.png');
3 h = ones(10,10)/10^2;
4 y = conv2(x,h, 'same');
5
6 figure(1), clf
7 image(x);
8 axis equal;
9 colormap gray; colorbar
10 title('Original [aka40]');
11
12 figure(2); clf
13 image(y)
14 axis equal; colormap gray; colorbar;
15 title("10x10 Blur [aka40]")
16 print -dpng IP1_EX5_Plot1.png
17
18
19 figure(4)
20 h = ones(2,50)/10^2;
21 y = conv2(x,h, 'same');
22 image(y)
23 axis equal; colormap gray; colorbar;
24 title("2x50 Blur [aka40]")
25
26 print -dpng IP1_EX5_Plot2.png
27 figure(3)
28 h = ones(50,2)/10^2;
29 y = conv2(x,h, 'same');
30 image(y)
31 axis equal; colormap gray; colorbar;
32 title("50x2 Blur [aka40]")
33
34 print -dpng IP1_EX5_Plot3.png
```

10.6 Exercise 6

```
1 clear
2 x = imread('coins.png');
3 h = ones(10,10)/10^2;
4 y = conv2(x,h, 'valid');
5
6 tc = linspace(0,1, 101);
7 xc = humps(tc);
8 deltatc = tc(2) - tc(1)
9 td = linspace(0,1,11);
10 xd = humps(td);
11 deltatd = td(2) - td(1)
12
13 figure(1), clf
14 image(x);
15 axis equal;
16 colormap gray; colorbar
17 title('Original [aka40]');
18
19 figure(2); clf
20 image(y)
21 axis equal; colormap gray; colorbar;
22 title("10x10 Blur [aka40]")
23 print -dpng IP1_EX6_Plot1.png
24
25
26 figure(4)
27 h = ones(2,50)/10^2;
28 y = conv2(x,h, 'valid');
29 image(y)
30 axis equal; colormap gray; colorbar;
31 title("2x50 Blur [aka40]")
32
33 print -dpng IP1_EX6_Plot2.png
34 figure(3)
35 h = ones(50,2)/10^2;
36 y = conv2(x,h, 'valid');
37 image(y)
38 axis equal; colormap gray; colorbar;
39 title("50x2 Blur [aka40]")
40
41 print -dpng IP1_EX6_Plot3.png
```

10.7 Exercise 7

```
1 clear
2 x = imread('coins.png');
3 gb5= [1 4 6 4 1; 4 16 24 16 4; 6 24 36 24 6; 4 16 24 16 4; 1 4 6 4 1]/256;
4
5 y1 = conv2(x,gb5, 'valid');
6 CoinEdgeX=[1 0 -1;1 0 -1; 1 0 -1];
7 CoinEdgeY=[1 1 1;0 0 0; -1 -1 -1];
8 figure(1),clf
9 image(y1);
10 axis equal;
11 colormap gray; colorbar
12 title('Coin Gaussian Blur [aka40]');
13 print -dpng IP1_EX7_Plot1.png
14
15
16 y2 = conv2(x, CoinEdgeX, 'valid');
17 figure(2),clf
18 imagesc(y2);
19 axis equal;
20 colormap gray; colorbar
21 title('Coin Vertical Edges [aka40]');
22 print -dpng IP1_EX7_Plot2.png
23
24 y3 = conv2(x, CoinEdgeY, 'valid');
25 figure(3),clf
26 imagesc(y3);
27 axis equal;
28 colormap gray; colorbar
29 title('Coin Horizontal Edges [aka40]');
30 print -dpng IP1_EX7_Plot3.png
31
32 y4= sqrt(y2.^2 + y3.^2);
33 figure(4),clf
34 imagesc(y4);
35 axis equal;
36 colormap gray; colorbar
37 title('Coin Edges [aka40]');
38 print -dpng IP1_EX7_Plot4.png
39
40
41 h = [-1 -1 -1; -1 8 -1; -1 -1 -1];
42 y5 = conv2(x,h, 'valid');
43 figure(5),clf
44 imagesc(y5);
45 axis equal;
46 colormap gray; colorbar
47 title('Alternate Coin Edges [aka40]');
48 print -dpng IP1_EX7_Plot5.png
```

10.8 Exercise 8

```
1 clear
2 img = imread('PM5544_with_non-PAL_signals.png');
3 figure(1); clf
4 image(img); axis equal
5 title('Original Test Card [aka40]')
6
7 print -dpng IP1_EX8_Plot0.png
8
9
10 BoxBlur = ones(21,21)/(21^2);
11 for k=1:3
12     y1(:,:,k) = conv2(img(:,:,k), BoxBlur, 'valid');
13 end
14 y1=uint8(y1);
15 figure(2); clf
16 image(y1);
17 axis equal; colormap gray
18 title('Box Blur Test Card [aka40]');
19 print -dpng IP1_EX8_Plot1.png
20
21 gb5= [1 4 6 4 1; 4 16 24 16 4; 6 24 36 24 6; 4 16 24 16 4; 1 4 6 4 1]/256;
22 for k=1:3
23     y2(:,:,k) = conv2(img(:,:,k), gb5, 'valid');
24 end
25 y2=uint8(y2);
26 figure(3); clf
27 image(y2);
28 axis equal; colormap gray
29 title('Gaussian Blur Test Card [aka40]');
30 print -dpng IP1_EX8_Plot2.png
31
32 SobelX=[-1 0 1; -2 0 2; -1 0 1];
33 for k=1:3
34     y3(:,:,k) = conv2(img(:,:,k), SobelX, 'valid');
35 end
36 y3=uint8(y3);
37 figure(4); clf
38 image(y3);
39 axis equal; colormap gray
40 title('Sobel Vertical Edges Test Card [aka40]');
41 print -dpng IP1_EX8_Plot3.png
42
43 SobelY=[1 2 1; 0 0 0; -1 -2 -1];
44 for k=1:3
45     y4(:,:,k) = conv2(img(:,:,k), SobelY, 'valid');
46 end
47
48 y4=uint8(y4);
49 figure(5); clf
50 image(y4);
51 axis equal; colormap gray
52 title('Sobel Horizontal Edges Test Card [aka40]');
53 print -dpng IP1_EX8_Plot4.png
54
55
```



```

56 Sobel = sqrt(SobelY.^2 + SobelX.^2);
57 for k=1:3
58     y5(:,:,k) = (y4(:,:,k).^2 + y3(:,:,k).^2);
59 end
60
61 y5=uint8(y5);
62 figure(6); clf
63 imagesc(y5);
64 axis equal; colormap gray
65 title('Sobel Edges Test Card [aka40]');
66 print -dpng IP1_EX8_Plot5.png

```