Assignment #3

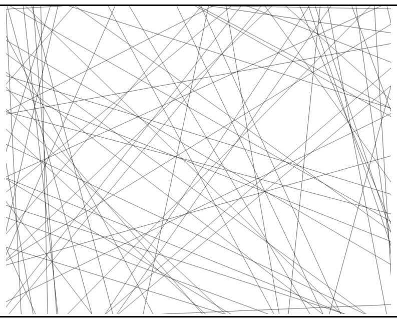
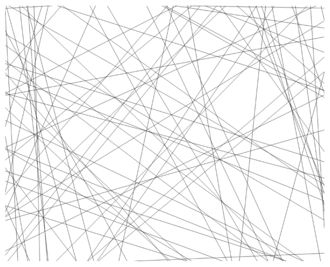
- Adwaith Venkataraman (Andrew ID: adwaithv)

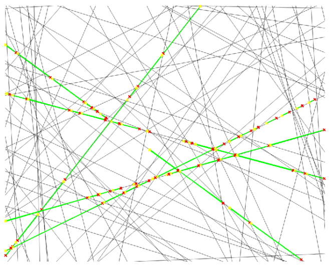
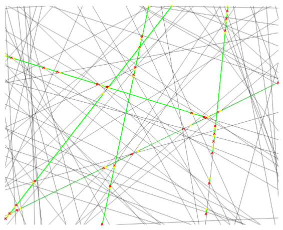
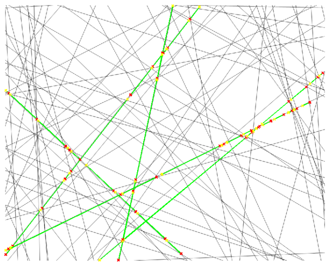
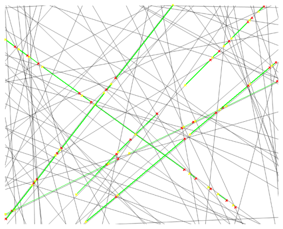
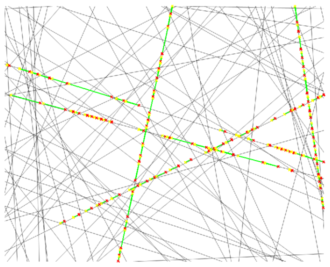
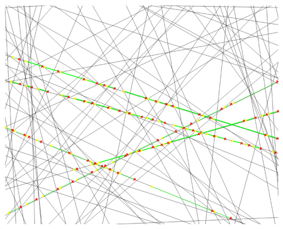
Question 1

Analysis:

1. The original image used is shown below. For purposes of simplicity, an image filled with lines was utilized.
2. Lines were detected from the original image and they were superimposed on the latter. For analyzing the line detection, variations were introduced in the image, like a change in the size, change in the resolution and changes in edge detection algorithms used.
3. The effect of change in image size was observed when it was noticed that the number of lines detected in the rescaled image was less in number than that detected in the original image. Also, the end points detected were lower in number.
4. When various edge detection algorithms were used, it was noticed that Roberts algorithm detects the most number of end points. This is because it uses the concept of detecting edges using the gradient of the image through differentiation.
5. Between Sobel and Canny algorithms, the number of end points detected in Canny was lesser in number. This is because the Canny algorithm uses the probabilistic method of detecting edges.
6. The resolution of the image was then modified (decreased) and was observed that the number of lines/end points detected was lesser in number that its counterparts. The variation in the value of pixels and the threshold resulted in a lesser efficient line detection.
7. The results obtained are tabulated below.

Results:

	
Original Image (1)	Rescaled Image (smaller)

	
Line Detection with Sobel sdge detection on Original Image	Line Detection with Sobel edge detection on Rescaled Image
	
Line Detection with Canny edge detection on Original Image	Line Detection with Canny edge detection on Rescaled Image
	
Line Detection with Roberts edge detection on Original Image	Line Detection with Roberts edge detection on Rescaled Image

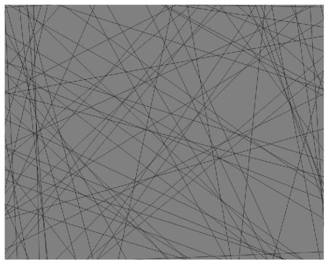
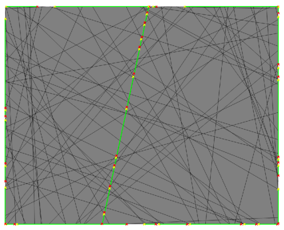
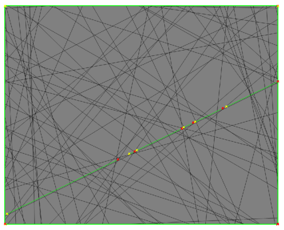
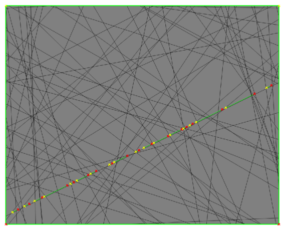
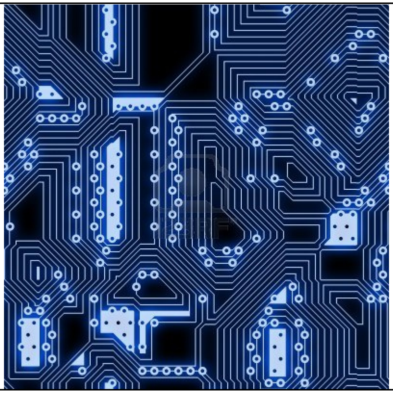
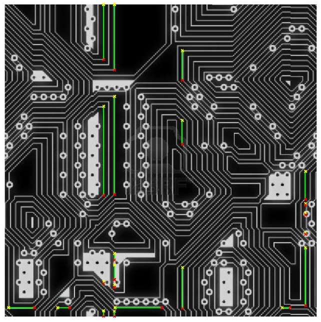
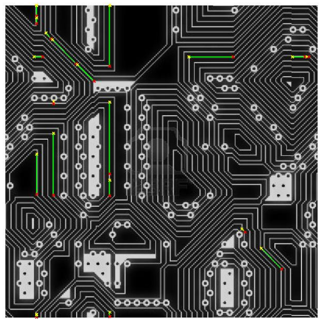
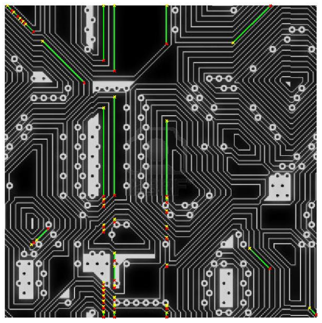
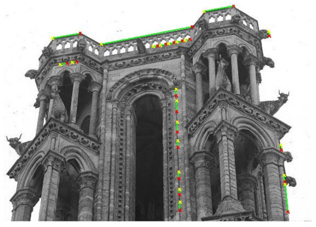
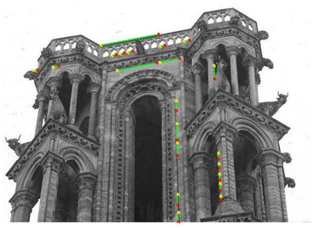
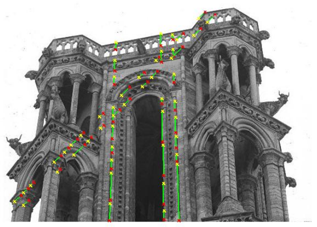

	

Image with a Lower Resolution	Lower Resolution Image with Sobel edge detection

	
Lower Resolution Image with Canny edge detection	Lower Resolution Image with Roberts edge detection
	
Original Image(2)	Line Detected Image with Sobel edge detection
	
Line Detected Image with Canny edge detection	Line Detected Image with Roberts edge detection
	
Original Image(3)	Line Detected Image with Sobel edge detection

	
Line Detected Image with Canny edge detection	Line Detected Image with Roberts edge detection

Matlab Code:

```
a = imread('lines3.jpg');
%a = imresize(b,0.5); %to rescale the image
I=rgb2gray(a);
BW = edge(I,'roberts');
[H,theta,rho] = hough(BW);
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
x = theta(P(:,2));
y = rho(P(:,1)); plot(x,y,'s','color','black');
lines = houghlines(BW,theta,rho,P,'FillGap',5,'MinLength',7);
imshow(I);
hold on;
max_len = 0;
for k = 1:length(lines)
xy = [lines(k).point1; lines(k).point2];
plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
% Plot beginnings and ends of lines
plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
% Determine the endpoints of the longest line segment
len = norm(lines(k).point1 - lines(k).point2);
if ( len > max_len)
max_len = len;
xy_long = xy; end
end
% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','red');
```

NOTE: The code snippet provided on Pages 28 and 29 in the document “Shape-Hough-V3.pdf” was utilized. However, certain modifications were imparted to it.

```
%to change the image resolution
a = imread('Lines_1024-819_72-72.png');
I=rgb2gray(a);
for j=1:819
    for i=1:1024
        I(j,i)=I(j,i)/2;
    end
end
imshow(I);
```

Question 2

Analysis:

1. A color image was read and stored as a 3-dimensional array.
2. The message to be encrypted was obtained from the user and the image with the hidden message is shown below.
3. The decimal value (ASCII) of each character from the string was obtained and was converted to binary format.
4. The length of the string was then obtained and stored.
5. Simultaneously, pixels of the 3-dimensional matrix of the image was read, converted to binary, and the least significant bit of that pixel was replaced by the corresponding bit value of the binary number, of the character.
6. Following this, the adjacent pixel was modified to store the next bit value of the same character.
7. By this method, in order to hide a character, when represented in an 8-bit binary format, requires 8 pixels of the image. Therefore, if the number of characters present in the string is say, 'n', then the minimum number of pixels required to encrypt the string is '8*n'.
8. Hence, in a 1024*768 color image, the maximum length of the string that can be encrypted is given as follows:

$$n_c = 1024 * 768 * {}^{3}/_{8}$$
$$= 294{,}912$$

9. To decode the hidden message from the image, each pixel of the image, starting from the origin was read, converted to binary and the least significant bit was stored in an array. The number of pixels read was based on the string length of the input string.
10. When the '8*n' bits were read and stored in an array, every 8 bits were converted into the ASCII decimal value, corresponding to each character.
11. This method is therefore a lossless form of encryption and decryption.
12. Once this was done, the resultant string was displayed.
13. The results are shown in the following table.

Results:

Original Image	Image with the hidden message

Matlab Command Window Output:

```
>> asg3_q2
Feed a message      hi! how are you?
Decoded Message:
hi! how are you?
```

Matlab Code:

```
b=imread('image1.jpg');
prompt='Feed a message      ';
c=input(prompt,'s'); %to take a string input
clen=length(c);
blen=length(b);
j=1;
m=1;
o=0;
%encoding the input string by replacing the LSB or pixels of the image
for k=1:clen
    str=dec2bin(c(k),8);
    for i=m:(m+7)
        if i>blen
            o=o+1;
            i=i-blen;
        end
        str1=dec2bin(b((j+(blen*o/blen)),i,1),8);
        str1(8)=str((mod((i-1),8)+1));
        b((j+(blen*o/blen)),i,1)=bin2dec(str1);
        i=i+1;
    end
    m=m+8;
    k=k+1;
end
imshow(b);

%decoding the hidden message from the original picture
j=1;
m=1;
o=0;
n=1;
for i=0:(clen-1)
    for k=1:8
        z=(i*8)+k;
        if (z>blen)
            z=z-blen;
            j=j+1;
        end
        b_str1=dec2bin(b(j,z,1),8);
        rstr((i+1),k)=b_str1(8);
        k=k+1;
    end
    i=i+1;
end
display('Decoded Message:');
for i=1:clen
    display(char(bin2dec(rstr(i,:))));
end
```