Assignment #2

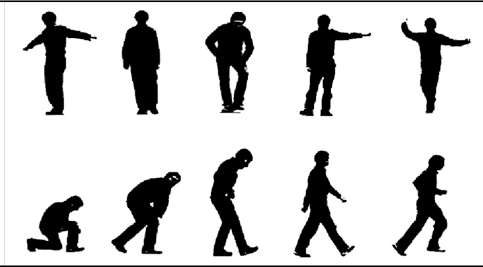
- Adwaith Venkataraman (Andrew ID: adwaithv)

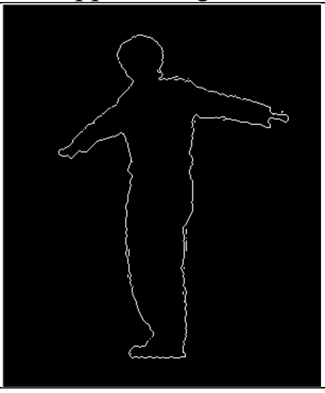
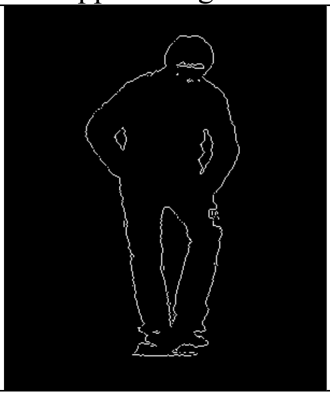
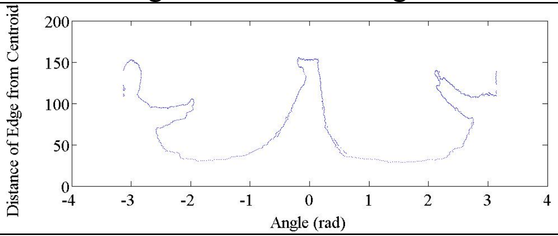
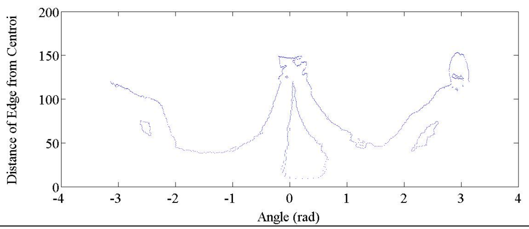
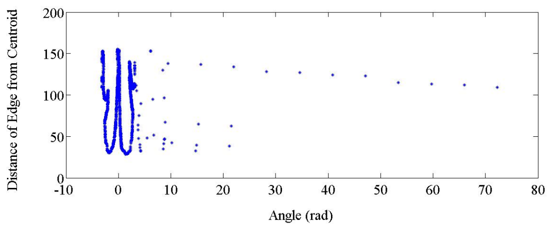
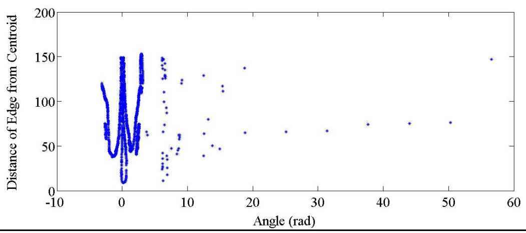
<u>Question 1</u>

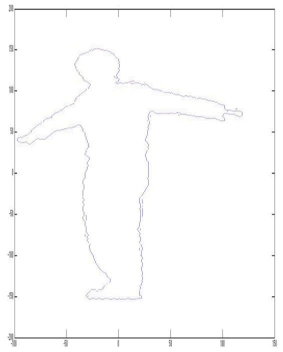
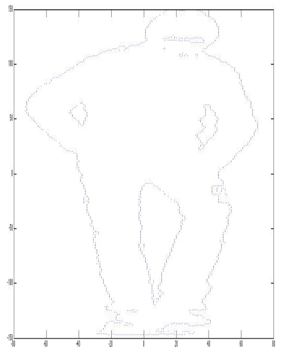
Analysis:

1. The given image was read, and two figures were cropped out of it, using 'Imcrop()'.
2. From the cropped image, the edges were detected, using 'edge()' and was stored in another variable.
3. Using nested for-loops, with the inner and outer loop variables ranging upto the X and Y dimension values of the cropped image, the Cartesian coordinates of each edge was converted to polar form, with respect to the centroid of the image.
4. The polar coordinates of all the edges was collectively stored in an array, where th[] represented the array of angle measurement of each edge, and r[] represented the array of distance of each edge from the centroid.
5. Also, a count of number of edges detected was kept count of in a variable, 'l'.
6. Now, when the polar plot was plotted, there seemed to be repetitive edges with the same angle, with respect to the x-axis.
7. Hence, when plotted in the polar form, multiple points with the same angle were found to align on the same vertical.
8. To prevent such an occurrence, the edge with a lesser distance i.e., 'r' value, was plotted on the primary domain [-pi,+pi], and those with greater 'r' values but with same 'theta' were plotted on adjacent domains [(2n-1)pi,(2n+1)pi] for all integers greater than 0.
9. Now the r-theta was plotted once again in the polar format.
10. During this, a flag value is initialized to zero, and a for-loop is run and the flag value is set to 1 if two edges with same theta is encountered.
11. It was found that there were no two edges with the same theta value.
12. To reconstruct the polygon from the polar plot, the 'theta' and 'r' values were read from the respective arrays, and the Cartesian form of each polar coordinate was obtained. The obtained coordinates were then plotted with the 'x' coordinate multiplied by '-1' to retrieve the original image.
13. The obtained results are tabulated in the following section.

Results:


Original Image

Cropped Image no. 1	Cropped Image no. 2
	
Edge Detected of Image 1	Edge Detected of Image 2
	
Polar Plot of Image 1, with redundant edges on the same theta	Polar Plot of Image 2, with redundant edges on the same theta
	
Polar Plot of Image 1, without redundant edges on the same theta	Polar Plot of Image 2, without redundant edges on the same theta

	
Retraced Polygon of Image 1	Retraced Polygon of Image 2

Matlab Code:

```
a=imread('figure1.jpg');
%imshow(a);
d=imcrop(a,[510 0 300 360]);
%d=imcrop(a,[0 0 300 360]);
%imshow(d);
i=edge(rgb2gray(d));
%imshow(i);

%to plot the polar form of the image.
l=1;
for m=1:360
    for n=1:300
        if i(m,n)==1
            [th(l) r(l)]=cart2pol(m-180,n-150,'bo');
            plot(th(l),r(l));
            if i(m,n)==1
                x=l;
            end
            l=l+1;
            hold on;
        end
    end
end
hold off;

%to re-plot the points with same theta, but onto a different domain.
for t=1:x
    %t=r+1;
    for k=(t+1):x
        if (th(k)==th(t))
            th(k)=th(k)+(2*pi); %the same theta, if there is more than
one r, then the theta value of the 2nd r must be added by 360.
        end
        k=k+1;
    end
    t=t+1;
end

%to plot the r-theta values without 2 points of same theta plotted on
```

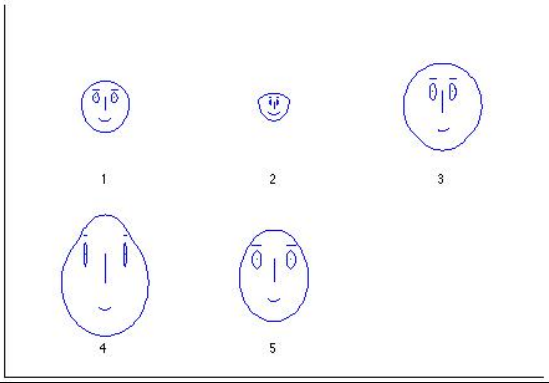
```
the
%same domain i.e, n(-pi) to n(pi).
for s=1:x
    plot(th(s),r(s),'b*');
    hold on;
    s=s+1;
end
hold off;

%to check if there are two points with the same theta value
y=0;
for t=1:x
    for k=(t+1):x
        if (th(k)==th(t))
            y=1;
            display(k); %to check the values of theta for which existed
more than one edge
        end
        k=k+1;
    end
    t=t+1;
end
display(y); %to display the number of instances redundancy of theta
occurs

%to retrieve the image from the polar plot
for i=1:x
    for j=1:x
        if(th(j)>(2*pi))
            th(j)=th(j)-(2*pi);
        end
    end
    [xc yc]=pol2cart(th(i),r(i));
    plot((yc),(-xc)); %to achieve the right orientation of the image,
as
    %original
    hold on;
end
hold off;

for i=1:x
    if(th(i)>(2*pi))
        display(i); %to check if any redundant theta values are present
    end
end
```

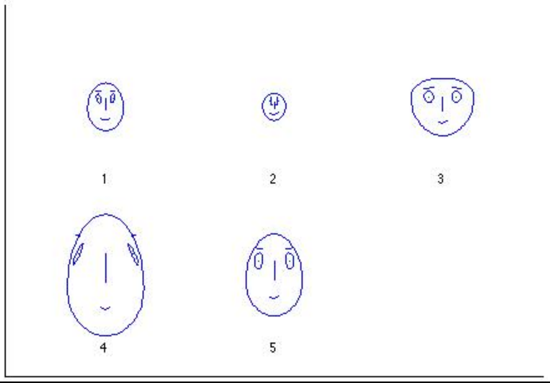
Question 2

Analysis:

1. The given data is fed into a 2-dimensional matrix.
2. The ‘glyphplot()’ function is invoked and the Chernoff faces are obtained.
3. Various combinations of the features are given as input and the variations in the Chernoff faces are observed.
4. The first combination of features was the default.

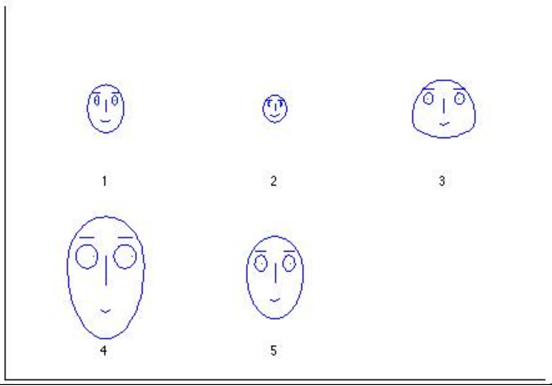
5. The second combination has variation in the shape of forehead, the width between the eyes, height of eyes and the angle of eyes.
6. The shape of the jaw, vertical position of the eyebrows and the direction of pupils were the adjusted features in the third case.
7. In the fourth case, the length of the arc was adjusted.
8. The vertical position of the mouth was altered in the fifth case.

	Pre_Break	Pre_Bed	Overall	OVOF	OTBG
DAY	Size Of Face	Forehead/jaw relative arc length	Shape of forehead	Shape of jaw	Width between eyes
1	110	125	117	135.0277	117.5
2	100	120	110	135.0277	110
3	130	110	120	135.0277	120
4	140	135	137	135.0277	137.5
5	125	130	127	135.0277	127.5
					
Observed Chernoff Faces (Output)					

	Pre_Break	Pre_Bed	Overall	OVOF	OTBG
DAY	Size Of Face	Shape of forehead	Width between eyes	Height of eyes	Angle of eyes
1	110	125	117	135.0277	117.5
2	100	120	110	135.0277	110
3	130	110	120	135.0277	120
4	140	135	137	135.0277	137.5
5	125	130	127	135.0277	127.5
					

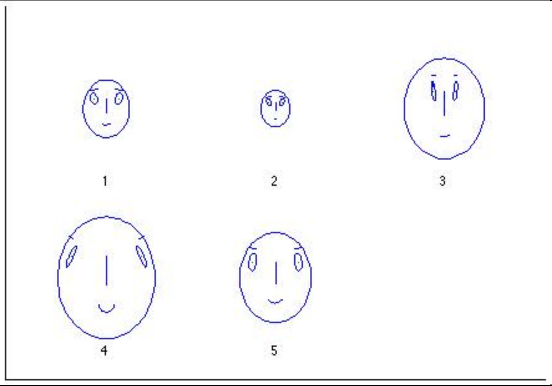
Observed Chernoff Faces (Output)

	Pre_Break	Pre_Bed	Overall	OVOF	OTBG
DAY	Size Of Face	Shape of jaw	Height of eyes	Vertical position of eyebrows	Direction of pupils
1	110	125	117	135.0277	117.5
2	100	120	110	135.0277	110
3	130	110	120	135.0277	120
4	140	135	137	135.0277	137.5
5	125	130	127	135.0277	127.5



Observed Chernoff Faces (Output)

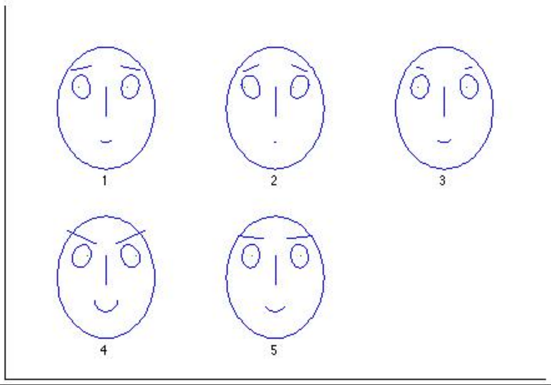
	Pre_Break	Pre_Bed	Overall	OVOF	OTBG
DAY	Size Of Face	Width between eyes	Angle of eyes	Direction of pupils	Mouth arc length
1	110	125	117	135.0277	117.5
2	100	120	110	135.0277	110
3	130	110	120	135.0277	120
4	140	135	137	135.0277	137.5
5	125	130	127	135.0277	127.5



Observed Chernoff Faces (Output)

	Pre_Break	Pre_Bed	Overall	OVOF	OTBG
DAY	Angle of eyes	Width of eyebrows	Direction of pupils	Vertical position of mouth	Mouth arc length

1	110	125	117	135.0277	117.5
2	100	120	110	135.0277	110
3	130	110	120	135.0277	120
4	140	135	137	135.0277	137.5
5	125	130	127	135.0277	127.5



Observed Chernoff Faces (Output)

Matlab Code:

```
x=[110,125,117,135.0277,117.5;
    100,120,110,135.0277,110;
    130,110,120,135.0277,120;
    140,135,137,135.0277,137.5;
    125,130,127,135.0277,127.5]; %the given data is fed into a 2-D
maatrix
F=[9 11 13 15 17]; %the attributes of the 'features' are set to a
particular combination
glyphplot(x,'Glyph','face','Features',F); %the chernoff faces are plot
```