# SHAPE PATTERN RECOGNITION USING EUCLIDEAN DISTANCE METHOD

Project Work Report Submitted

in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in

ELECTRICAL AND ELECTRONICS ENGINEERING

**By**

**Rushil Anirudh**
**(06EE49)**

**Srinivas L Naik**
**(06EE59)**

**Sunil Saraswat Swain**
**(06EE63)**

*Under the Guidance of*

**Dr. Ashvini Chaturvedi**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGYKARNATAKA SURATHKAL, MANGALORE – 575 025**

**April, 2010**

**DECLARATION**

By the B.Tech Student(s)

We hereby declare that the Project Work Report entitled **— "Shape Pattern Recognition using Euclidean Distance Method"** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** for the award of the degree of **Bachelor of Technology in Electrical and Electronics Engineering** is a bonafide report of the work carried out by us. The material contained in this Project Report has not been submitted to any University or Institution for the award of any degree.

1. Rushil Anirudh (06EE49)
2. Srinivas L Naik (06EE59)
3. Sunil Saraswat Swain (06EE63)

**Department of Electrical & Electronics Engineering**

Place: NITK, Surathkal

Date:

## CERTIFICATE

This is to certify that the project entitled **“Shape Pattern Recognition using Euclidean Distance Method”** submitted by**:**

1. Rushil Anirudh (06EE49)
2. Srinivas L Naik (06EE59)
3. Sunil Saraswat Swain (06EE63)

as the record of the work carried out by them, is accepted as the B.Tech Project Work Report submission in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in Electrical & Electronics Engineering.

| | |
|---|---|
| **Guide** | **Head of Department** |
| **Dr. Ashvini Chaturvedi** | **Prof.P Durai Kannu** |

Place: NITK, Surathkal

Date:

## ACKNOWLEDGEMENT

The successful completion of our project gives us an opportunity to convey heartfelt regard to each and everyone who has been instrumental in shaping up the final outcome of this project.

We would like to thank Dr. Ashvini Chaturvedi, Department of Electrical and Electronics Engineering, NITK - Surathkal for his able guidance and encouragement during our project work which was of great help in the successful completion of the project.

We thank Dr. P.Duraikannu, Professor and HOD, Department of Electrical and Electronics Engineering, NITK Surathkal for his support and help to utilize the resources available at Department Lab.

Finally we would like to thank everyone who was a great source of help during various stages of the project.

# CONTENTS

## **Abstract**

*Shape recognition has been a problem that has been approached from a large number of perspectives, as discussed later. It has been seen as a critical component in man's effort to emulate his own intelligence. Shape recognition poses special problems – in terms of not just distance differences and hence size variation, but also in terms of rotational variance. Even a few degrees of difference has been known to cause recognition difficulties. This project is a bid to develop a fresh algorithm that can solve the problems of size and shape variant difficulties. One of the most classical applications of shape recognition has been hand gesture recognition and it will be exactly this problem that we shall try and solve using our proposed algorithm. Given ahead are known approaches to the problem, the logical flow of our approach, the MATLAB code, the outputs and results along with a compilation of possible improvements and final conclusions. A reference for future works has also been provided.*

**Base Paper Review :**

The following was used as a base paper:

E. Sanchez-Nielsen, L. Antón-Canalis, M. Hernan-dez-Tejera, ***Hand gesture recognition for human-machine interaction***, Journal of WSCG, Vol.12,No.1-3 (February 2003).

The aim of this paper is the proposal of a real time vision system for its application within visual interaction environments through hand gesture recognition, using general-purpose hardware and low cost sensors, like a simple personal computer and an USB web cam, so any user could make use of it in his office or home. The most important part of the recognition process is a robust shape comparison carried out through a Hausdorff distance approach, which operates on edge maps. The use of a visual memory allows the system to handle variations within a gesture and speed up the recognition process through the storage of different variables related to each gesture.

Initially the segmentation was done using Hue-Saturation-Intensity (**HSI**) parameters. This was followed by identification of the biggest binary linked object (blob). The sample image is then compared to the images in the visual memory system for a good match. The matching algorithm used here is the Hausdorff distance method.

Hausdorff Distance ranks each point of an array A based on its distance to the nearest point in a second array *B* and then uses the largest ranked such point as the measure of distance.

**Hausdorff Distance (HD):**

$H(A, B) = max(h(A,B),h(B, A))$

where,

$h(A,B)=max_{a\epsilon A}\ min_{b\epsilon B}\ ||a-b||$

The function *h (A,B)* is called the *directed Hausdorff* distance from set *A to B.*

The HD algorithm is complicated for Shape Matching and results can be achieved with reasonable accuracy even by considering a few selected points instead of the entire image. Another important drawback of this algorithm is that it fails to incorporate slight changes in position of the object in consideration.

## APPROACHES

1) The shape matching processes can be classified into two levels, namely the global shape matching and local shape matching. Global shape matching refers to the comparison of the overall shape of an object which deals with shape similarity. Local shape matching refers the matching of a pair of points in two similar shapes which deals with feature correspondence.

   a) Common shape matching methods include :

      1. **Shape distribution**: Reference [11] applied the shape distribution method to meshed and faceted representations and used the membership classification distribution as the basis for a shape classification histogram.

      2. **Shape context:** Reference [4] used shape context method to represent each point in the object as the relative vector of each point on an object relative to all remaining points, in an L2 Euclidean metric. This leaded to an efficient representation of characteristics histograms useful for shape comparison. This method gives reasonable good performance in shape matching and feature correspondence but it suffers from high computational cost.

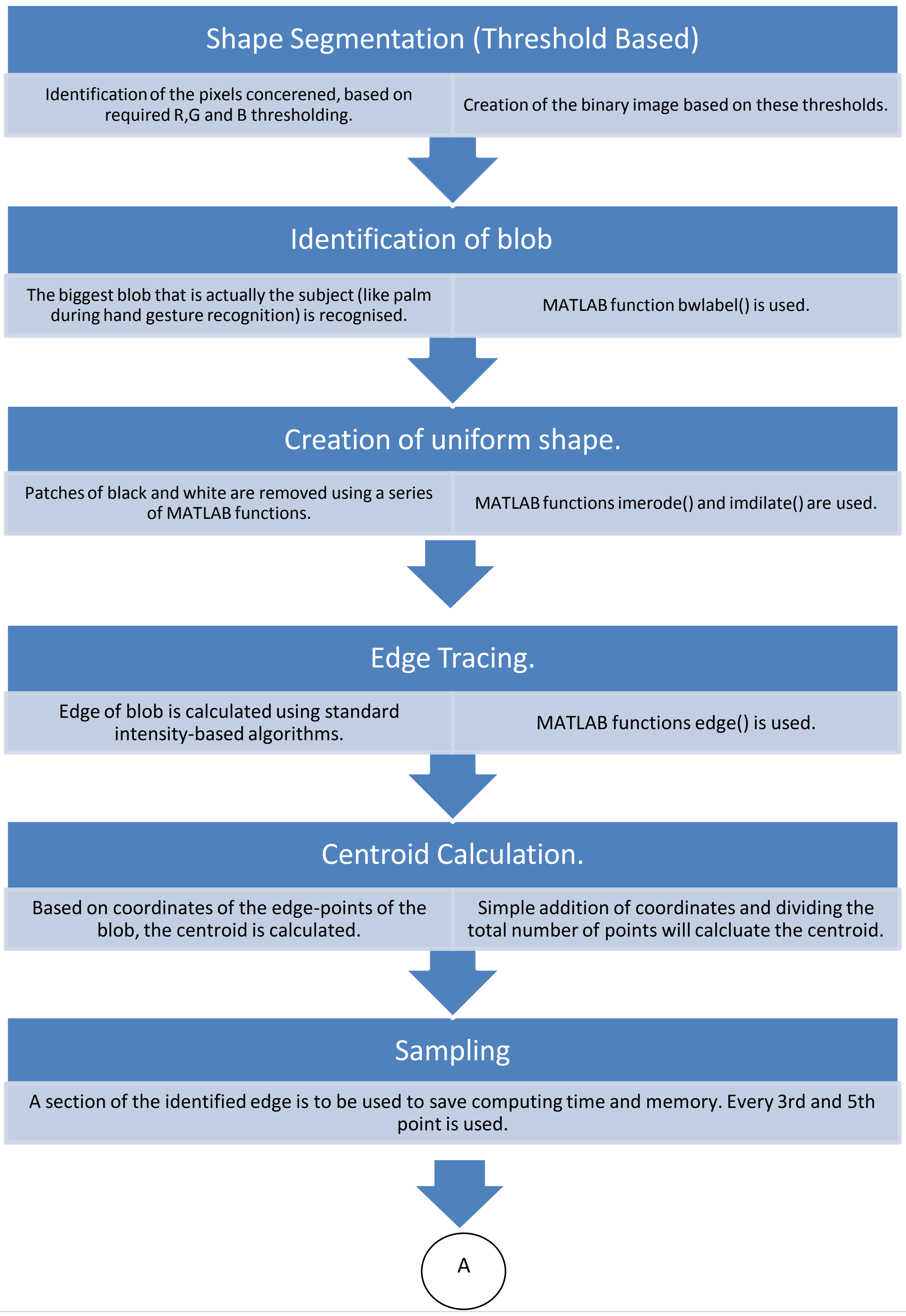
      3. **Curvature scale space** : The curvature scale space method in [17, 18] took advantage of connectivity between contour points and the shape of object is represented by a graph of the parametric positions of the curvature extreme points of extreme curvature on the contour, however only curvature extreme points can be extracted.

      4. **The Euclidean Distance Method**: The centroid of the image and its distance of boundary is calculated. These distances are then compared with the existing database images and the best match is found.

      5. **Vector Method**: An array of distance as well as angles between the centroid and the boundary point are calculated. A few selected points from the incoming image are compared with the existing database.

      6. **Shifting of the Origin Method**: A new set of coordinates are calculated with respect to the Centroid position.

## Flow Diagram



**Shape Segmentation (Threshold Based)**

Identification of the pixels concerened, based on required R,G and B thresholding.

Creation of the binary image based on these thresholds.

**Identification of blob**

The biggest blob that is actually the subject (like palm during hand gesture recognition) is recognised.

MATLAB function bwlabel() is used.

**Creation of uniform shape.**

Patches of black and white are removed using a series of MATLAB functions.

MATLAB functions imerode() and imdilate() are used.

**Edge Tracing.**

Edge of blob is calculated using standard intensity-based algorithms.

MATLAB functions edge() is used.

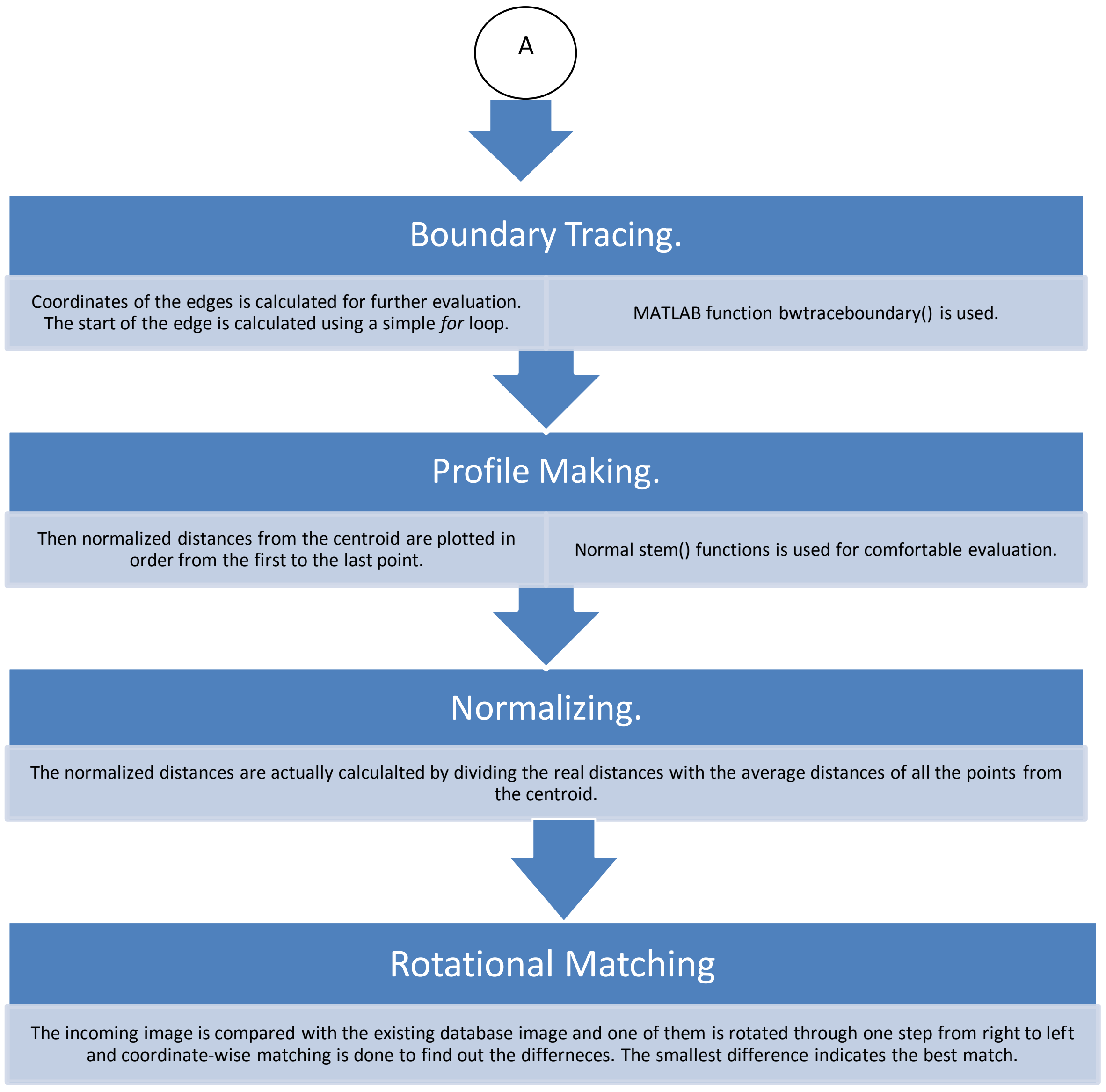
**Centroid Calculation.**

Based on coordinates of the edge-points of the blob, the centroid is calculated.

Simple addition of coordinates and dividing the total number of points will calcluate the centroid.

**Sampling**

A section of the identified edge is to be used to save computing time and memory. Every 3rd and 5th point is used.



**Boundary Tracing.**

Coordinates of the edges is calculated for further evaluation. The start of the edge is calculated using a simple *for* loop.

MATLAB function bwtraceboundary() is used.

**Profile Making.**

Then normalized distances from the centroid are plotted in order from the first to the last point.

Normal stem() functions is used for comfortable evaluation.

**Normalizing.**

The normalized distances are actually calculalted by dividing the real distances with the average distances of all the points from the centroid.

**Rotational Matching**

The incoming image is compared with the existing database image and one of them is rotated through one step from right to left and coordinate-wise matching is done to find out the differneces. The smallest difference indicates the best match.

## Methodology:

I. **Segmentation:** This can be achieved in several ways

   a. **RGB Values** – Here a fixed range of Red/Green/Blue values of the pixels are conveniently chosen to distinguish the object from the background.

   b. **Normalised RGB Values** – Depending upon the ratio of one of the colour components, segmentation is done.

   c. **HSI/HSV** – The Hue, Saturation and Intensity/Value decide the segmentation.

   In this experiment, the method using RGB Values is used for segmentation.

II. **Binary Linked Object Identification** – The binary equivalent of the object is created by thresholding the incoming image. The biggest blob is found using MATLAB® function `bwlabel`(C,Appendix 2)

III. **Noise Removal** – Since a few pixels in the image may not fall in the range of the chosen RGB values, it accounts as noise. This can be reduced greatly by using standard MATLAB® functions like `imdilate` (A ,Appendix 2) and `imerode` (B, Appendix 2). While the former enhances the object or the foreground, the latter enhances the background.

IV. **Edge Detection** – The MATLAB® function `edge` (E, Appendix 2) outlines the binary image. The boundary coordinates of the outline are calculated using MATLAB® function `bwboundarytrace` (D, Appendix 2). Using a known edge point, this function returns an array with the coordinates of the boundary.

V. **Calculation of Euclidean Distances**

   a. The position of the centroid is calculated using the following formula:
      **$C_X = \Sigma x_i / n$ $C_Y = \Sigma y_i / n$**
      Where, n-*Number of points* and x,y are the *coordinates*

   b. The distance of the edge point from the center is calculated using the standard Euclidean formula which is : $\sqrt{[(x - Cx)^2 + (y - Cy)^2]}$

   c. Once the distances are calculated, the values are normalized by dividing the all the elements with the average value of the array. This takes care of positioning of the object with respect to the camera.

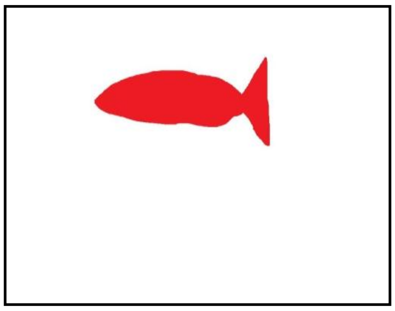
VI. **Pattern Matching** – Each image in the database is associated with a unique set of normalized Euclidean Distances. The corresponding array of the incoming image is compared with the existing arrays to find a best match. The best match is defined as the array which yields the least difference.

VII. **Rotational invariance-** Since the angle of the object may vary with every incoming image the program is made rotational invariant. Every point of the object is shifted in a cyclic manner and the corresponding image is then compared with the stored images respectively. At the point of matching, the difference between the rotated image and the stored image is the least value. This is achieved by using MATLAB® function `circshift` (F, Appendix 2). Circshift shifts the column of the matrix cyclically. The differences are stemmed and the point of least difference is the best possible match.

## Summary

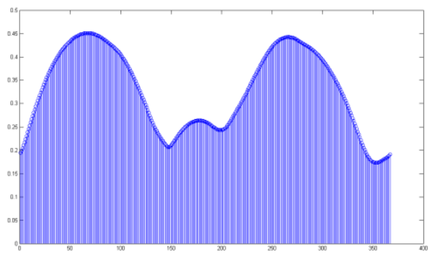
A database is created using a set of standard pictures. The given image is first converted to a binary image by thesholding. Based on the pixel values (falling in a fixed range of R, G and B) thresholding is done. The biggest blob is found using `bwlabel()`(C, Appendix 2). After using matlab functions including `imdilate`(A, Appendix 2) and `imerode`(B, Appendix 2), the edge of the blob is found. The centroid of the image is got by the summation of the coordinates of the individual points. The boundary is located using a MATLAB function `bwtraceboundary` (D, Appendix 2) and a known edge point. Sampling is done to increase the time and memory efficiency. The Euclidean distance of each point from the calculated centroid is evaluated and normalization is done by dividing the individual distance with the average of the net distance. To incorporate rotational invariance, the given image is rotated and subtracted from the standard images. The standard image which shows the least possible difference is taken as the best match.

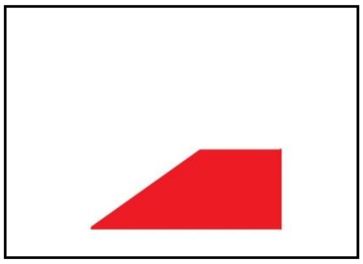
## Results

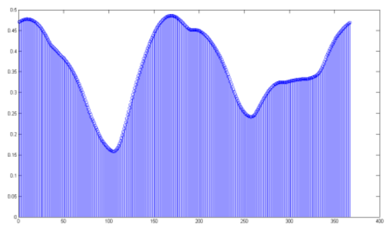


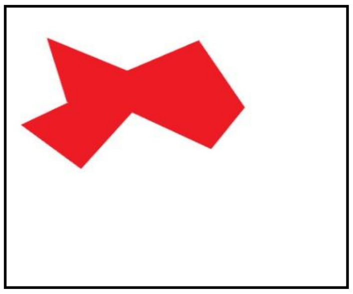
Incoming image

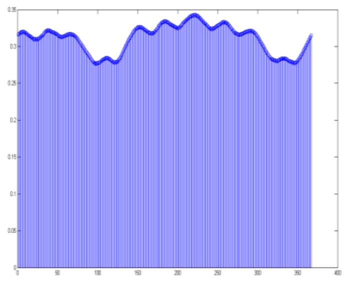
Stemmed Graphs showing comparison with the existing images

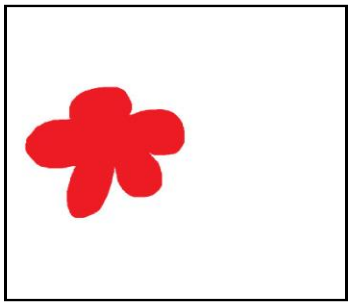


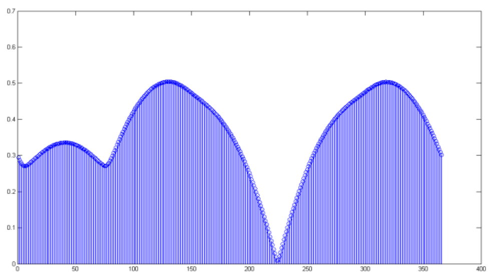




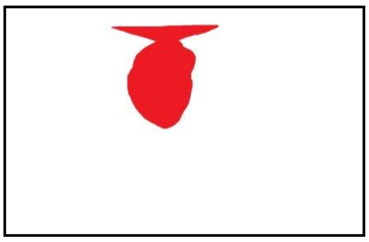


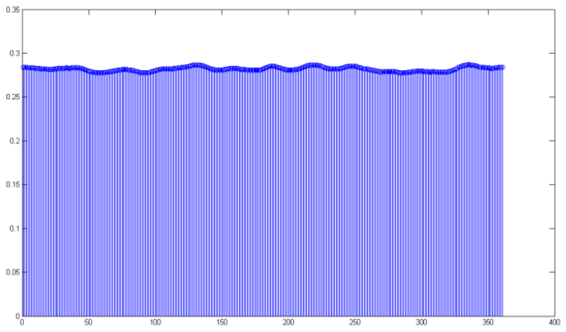


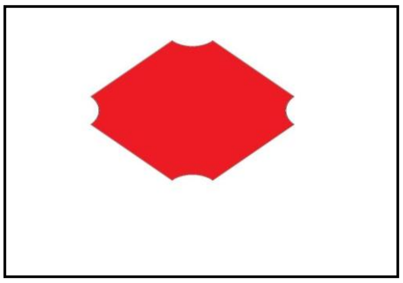




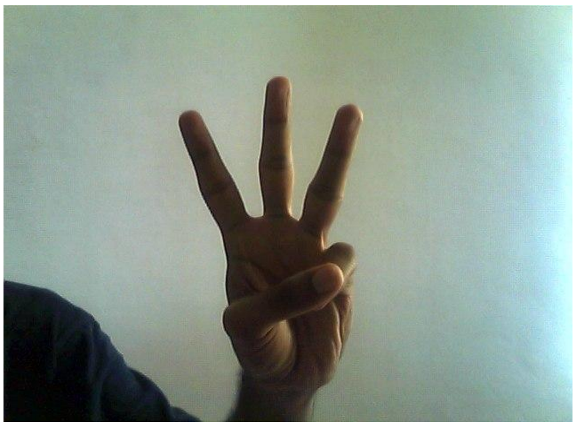
**Best Match**

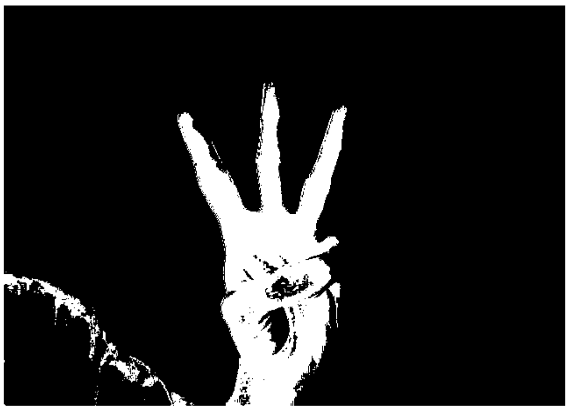


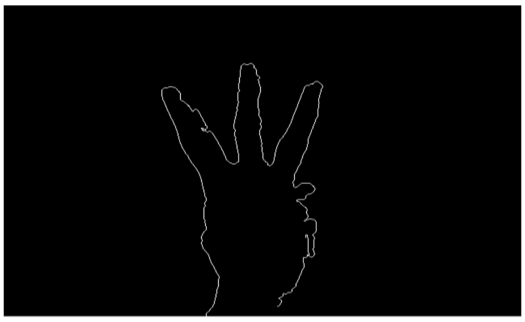




## Algorithm Usage in Future Applications


Initial Image from Webcam (RGB)


Colour Segmentation based on on-the-spot readings.



Biggest blob from the previous image is found and the edges detected to yield the boundary that is traced and a profile made

## Conclusions

A reasonable accuracy can be obtained with the Euclidean distance method by using a finite set of points. The position and the orientation of the object does not affect the working of the algorithm. Size invariance was achieved through normalisation to reduce the error caused due to the positioning of the object. The same image irrespective of its size always yields the same result.

The change in the angle of the object is also achieved by circular shift. The given image was rotated through all the points and compared with the images. Hence the best possible match could easily be found corresponding to a particular image. Rotational invariance was one of the major setbacks of the Hausdorff method.

The simplicity and easy to implement nature of the algorithm gives it an edge over the Hausdorff method.

## Applications and future work

1. Since the algorithm is generic and simple to implement, it can be modified to be used in all applications requiring pattern recognition like – gesture recognition etc.
2. Improved segmentation by using probabilistic models for variations in light.
3. Rotational invariance can also be achieved using the vector approach.

## Appendix 1

### CODE

```
clc
clear
warning off
t = ['trapedown']
title = [t '.jpg'];
    bin(:,:) = 0;
    im = imread(title);
    r = size(im,1);
    c = size(im,2);
    im1 = im;
    im1(:,:,:) = 0;
    bin(r,c) = 0;
    for i = 1:r
        for j = 1:c
            if  (im(i,j,1)>200  &&  im(i,j,1)<250  &&  im(i,j,2)>10  &&
im(i,j,2)<40 && im(i,j,3)>10 && im(i,j,3)<50)
                im1(i,j,1) = im(i,j,1);
                im1(i,j,2) = im(i,j,2);
                im1(i,j,3) = im(i,j,3);
                bin(i,j) = 1;
            end
        end
    end

    [L x] = bwlabel(bin,8);
    numb(x) = 0;

    for i = 1:r
        for j = 1:c
            for k = 1:x
                if L(i,j) == k
                    numb(k) = numb(k) + 1;
                end
            end
        end
    end


    [qaz wsx] = max(numb);

    for i = 1:r
        for j = 1:c
            if L(i,j) == wsx
                bin(i,j) = 1;
            else
                bin(i,j) = 0;
            end
        end
    end
```

```
rsum = 0;
csum = 0;
num = 0;

for i = 1:r
    for j = 1:c
        if bin(i,j) == 1
            rsum = rsum + i;
            csum = csum + j;
            num = num + 1;
        end
    end
end

se = strel('disk',3,0);
bin = imdilate(bin,se);
bin = imerode(bin,se);
bin = imdilate(bin,se);
bin = edge(bin);

for i = 1:r
    for j = 1:c
        if bin(i,j) == 1
            start_r = i;
            start_c = j;
        end
    end
end

B = bwtraceboundary(bin,[start_r start_c],'N');
rsample = B(:,1);
csample = B(:,2);

r_centre = rsum/num;
c_centre = csum/num;
bin(floor(r_centre),floor(c_centre)) = 128;
qaz = size(rsample,1);




rsample = rsample(1:3:end);
csample = csample(1:3:end);

n_points = size(rsample,1);
bina = bin;
bina(:,:) = 0;
for i = 1:size(rsample,1)
    bina(rsample(i,1),csample(i,1)) = 255;
end

imshow(bina)
```

```
    for i = 1:n_points
        euc_dist(i) = sqrt( (rsample(i) - r_centre)^2 + (csample(i)
- c_centre)^2 );
    end
    euc_dist = euc_dist./(sum(euc_dist)/numel(euc_dist));

    for i = 1:n_points
        angle(i)  =  atan2(  rsample(i)  -  r_centre,  csample(i)  -
c_centre );
    end

%     imtool(bin);
    bin = uint8(bin);
%      figure
    % stem(euc_dist);
```

## Appendix 2

A. Imdilate – IMDILATE Dilate image.

IM2 = IMDILATE(IM,SE) dilates the grayscale, binary, or packed binary image IM, returning the dilated image, IM2. SE is a structuring element object, or array of structuring element objects, returned by the STREL function. If IM is logical (binary), then the structuring element must be flat and IMDILATE performs binary dilation. Otherwise, it performs grayscale dilation. If SE is an array of structuring element objects, IMDILATE performs multiple dilations, using each structuring element in SE in succession.

Examples
--------
Dilate the binary image in text.png with a vertical line:

```
originalBW = imread('text.png');
se = strel('line',11,90);
dilatedBW = imdilate(originalBW,se);
figure, imshow(originalBW), figure, imshow(dilatedBW)
```

Dilate the grayscale image in cameraman.tif with a rolling ball:

```
originalI = imread('cameraman.tif');
se = strel('ball',5,5);
dilatedI = imdilate(originalI,se);
figure, imshow(originalI), figure, imshow(dilatedI)
```

Determine the domain of the composition of two flat structuring elements by dilating the scalar value 1 with both structuring elements in sequence, using the 'full' option:

```
se1 = strel('line',3,0);
se2 = strel('line',3,90);
composition = imdilate(1,[se1 se2],'full')
```

B. Imerode

M2 = IMERODE(IM,SE) erodes the grayscale, binary, or packed binary image IM, returning the eroded image, IM2. SE is a structuring element object, or array of structuring element objects, returned by the STREL function. If IM is logical and the structuring element is flat, IMERODE performs binary erosion; otherwise it performs grayscale erosion. If SE is an array of structuring element objects, IMERODE performs multiple erosions of the input image, using each structuring element in succession.

Examples
--------
Erode the binary image in text.png with a vertical line:

```
      originalBW = imread('text.png');
      se = strel('line',11,90);
      erodedBW = imerode(originalBW,se);
      figure, imshow(originalBW)
      figure, imshow(erodedBW)
```

Erode the grayscale image in cameraman.tif with a rolling ball:

```
   originalI = imread('cameraman.tif');
      se = strel('ball',5,5);
      erodedI = imerode(originalI,se);
      figure, imshow(originalI), figure, imshow(erodedI)
```

C. Bwlabel

L = BWLABEL(BW,N) returns a matrix L, of the same size as BW, containing labels for the connected components in BW. N can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects; if the argument is omitted, it defaults to 8. The elements of L are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labeled 1 make up one object, the pixels labeled 2 make up a second object, and so on.
[L,NUM] = BWLABEL(BW,N) returns in NUM the number of connected objects found in BW.

Example:

```
 BW = logical([1  1  1  0  0  0  0  0
              1  1  1  0  1  1  0  0
              1  1  1  0  1  1  0  0
              1  1  1  0  0  0  1  0
              1  1  1  0  0  0  1  0
              1  1  1  0  0  0  1  0
              1  1  1  0  0  1  1  0
              1  1  1  0  0  0  0  0]);
L = bwlabel(BW,4);
[r,c] = find(L == 2);
```

D. Bwtraceboundary

B = BWTRACEBOUNDARY(BW,P,FSTEP) traces the outline of an object in a binary image BW, in which nonzero pixels belong to an object and 0-pixels constitute the background. P is a two-element vector specifying the row and

column coordinates of the initial point on the object boundary. FSTEP is a string specifying the initial search direction
for the next object pixel connected to P. FSTEP can be any of the following strings: 'N','NE','E','SE','S','SW','W','NW', where N stands for north, NE stands for northeast, etc. B is a Q-by-2 matrix, where Q is the number of boundary pixels for the region. B holds the row and column coordinates of the boundary pixels.

```
BW = imread('blobs.png');
       imshow(BW,[]);
       s=size(BW);
       for row = 2:55:s(1)
         for col=1:s(2)
           if BW(row,col),
             break;
           end
         end

         contour  =  bwtraceboundary(BW,  [row,  col],  'W',  8,
50,...
                                 'counterclockwise');
         if(~isempty(contour))
           hold                                              on;
plot(contour(:,2),contour(:,1),'g','LineWidth',2);
           hold on; plot(col, row,'gx','LineWidth',2);
         else
           hold on; plot(col, row,'rx','LineWidth',2);
         end
       end
```

E. Edge

EDGE takes an intensity or a binary image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

```
I = imread('circuit.tif');
        BW1 = edge(I,'prewitt');
        BW2 = edge(I,'canny');
        figure, imshow(BW1)
        figure, imshow(BW2)
```

F. Circshift

B = CIRCSHIFT(A,SHIFTSIZE) circularly shifts the values in the array A by SHIFTSIZE elements. SHIFTSIZE is a vector of integer scalars where the N-th element specifies the shift amount for the N-th dimension of array A. If an element in SHIFTSIZE is positive, the values of A are shifted down (or to the right). If it is negative, the values of A are shifted up (or to the left).

Examples:

```
        A = [ 1 2 3;4 5 6; 7 8 9];
        B = circshift(A,1) % circularly shifts first dimension
values down by 1.
        B =     7     8     9
                1     2     3
                4     5     6
        B = circshift(A,[1 -1]) % circularly shifts first
dimension values
                              % down by 1 and second
dimension left by 1.
        B =     8     9     7
                2     3     1
                5     6     4
```