ARTIFICIAL INTELLIGENCE CSE3013

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

TOPIC:

SIGN LANGUAGE ALPHABET RECOGNITION

Submitted to:

Dr. Priyadarshini J

Members:

AKASH (15BCE1369)

NIDHI JOHNSON (15BCE1157)

KUSHAGRA SRIVASTAVA (15BCE1064)

ABSTRACT:

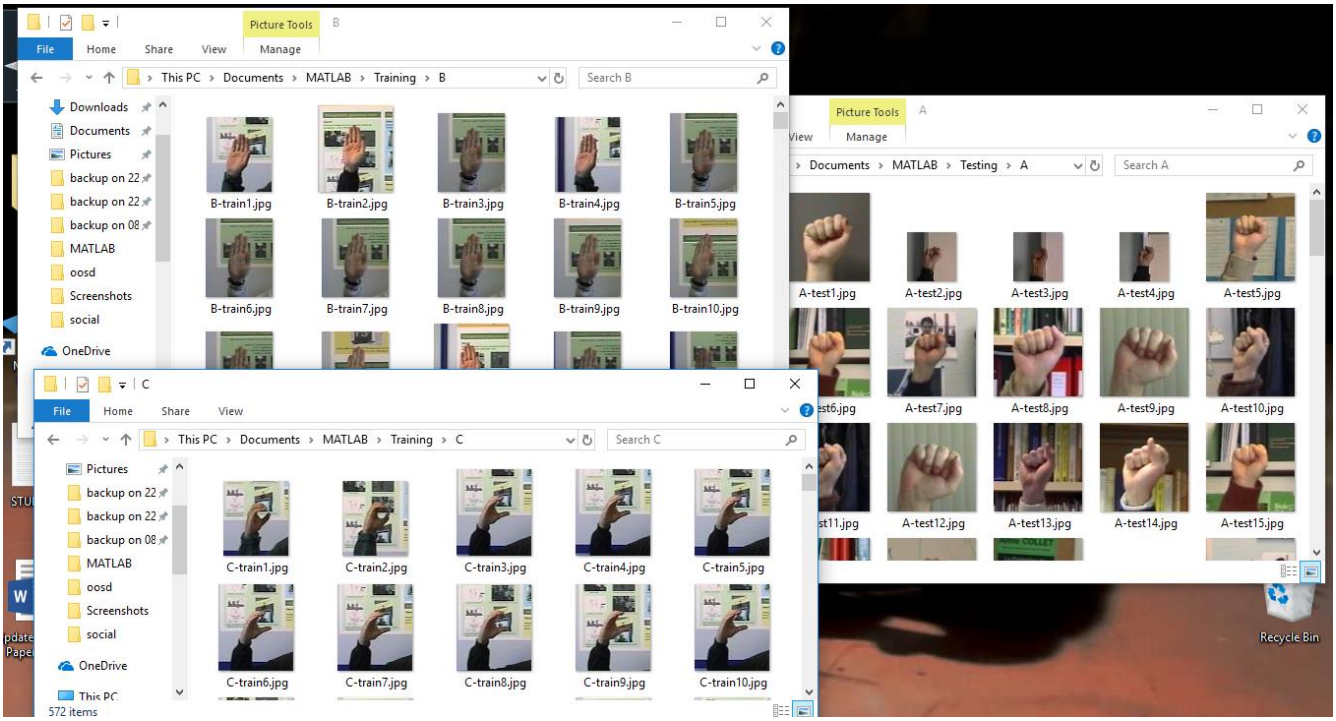
Most deaf children are born into hearing families and the majority do not begin their Sign Language learning before age 4-6 in school. As children learning Sign Language from infancy show higher language proficiency than those who do not, it underlines the importance of hearing parents learning how to sign. Despite the increased commercial and academic interest in gesture recognition over the last two decades, there are currently no widespread system to aid Sign Language learners. This project proposes a gesture recognition system for both fixed and gesture based signs by either detecting through fed in images or through a live motion video. Additionally, to support local variations and nuances in signs, the system offers tutors the ability to build unique datasets.

KEYWORDS:

- SIGN LANGUAGE
- GESTURE RECOGNITION
- SIFT
- KNN
- SVM
- PCA
- FLD

INTRODUCTION:

We have a database containing around 435 images of different hand sign gestures from different angles and distances as well for each alphabet so that we are able to achieve most accurate results.



We have developed our project on MATLAB. Matlab-image-processing toolbox is used for this purpose. The first algorithm which we have used is the SIFT algorithm.
The **scale-invariant feature transform (SIFT)** is an algorithm in computer vision to detect and describe local features in images. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.
SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches.

Another algorithm used is KNN. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. The k-NN algorithm classifies unknown data points by finding the *most common class* among the **k-closest examples.** Each data point in the *k* closest examples casts a vote and the category with the most votes wins.

The next concept used is SVM. SVM can be used to optimize classification of images (or subimages, for segmentation). SVM stands for Support Vector Machine. An SVM might be used to perform image classification for example, given an input image it decides whether it's a cat or a dog. The image, before being input into the SVM might have gone through some image processing filters so that some features might be extracted such as edges, color and shape.

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of distinct principal components is equal to the smaller of the number of original variables or the number of observations minus one. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. PCA represents the data in a new coordinate system in which basis vectors follow modes of greatest variance in the data. Thus, new basis vectors are calculated for the particular data set.

Lastly we are using LDA. Linear discriminant analysis (LDA) or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data.[4] LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from

factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made. . Linear discriminant analysis is primarily used here to reduce the number of features to a more manageable number before classification. Each of the new dimensions is a linear combination of pixel values, which form a template.

LITERATURE SURVEY

The human hand is a highly deformable object with many degrees of freedom . To recognise the sign language, the input image has to undergo many processes through which the required information will be collected to identify the gesture. There are many methods in each process. The image can be acquired by using a fixed camera or by using multiple cameras. After obtaining the image it has to be segmented where only the required information will be collected. After segmenting the image the features will be extracted from the segmented region in order to recognise the gesture. And the extracted image is studied in depth using the learning algorithms. Each of these modules has several techniques associated with them. This chapter discusses the techniques that have been implemented in gesture recognition systems, their pros and cons and also presents an idea of how they could be used in ISL recognition

2.1 SURVEY OF IMAGE ACQUISITION TECHNIQUES

The image acquisition is the first stage of any computer vision based technology. The processing can be done on the image only when the image has been obtained. If the image has not been acquired properly, the satisfactory output will not be obtained even after so many image processing techniques (i.e.) if the gesture has not been acquired properly from the input, it cannot be recognised. So the image acquisition plays a pivotal role in computer vision based system. The two dimensional image is usually described in terms of spatial co ordinates. An image is said to be digitized after spatial sampling and intensity quantization. The input can be obtained by many methods. It can be done be using:

2.1.1. Single Camera

An important factor in any computerized system is its viability to reach the common man, which boils down to the affordability of the built system. In computer vision 9 system using a single camera to acquire input is an effective way to bring down the cost.

2.1.2. Multiple Cameras

Multiple cameras are used to acquire more than one perspective of an object. Usually multiple cameras are used in computer vision applications which require high precision. The most important feature that is extracted using multiple cameras is the depth of the objects. Response time is very important in the system being developed because ISL recognition has to happen at real time. Hence going for multiple cameras is a disadvantage. Apart from the using a single or multiple cameras for image acquisition, the user sometimes is required to use some additional hardware to felicitate proper image acquisition. There are 2 different approaches used in several gesture based technologies. • Data glove or cyber glove. • Computer vision based human computer interaction using hand gesture.

2.1.3. Glove based hand gesture recognition

The Cyber glove II (a product of Cyber Glove systems, USA) is a fully instrumented glove that provides up to 22 high-accuracy joint-angle measurements. It uses proprietary resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data.

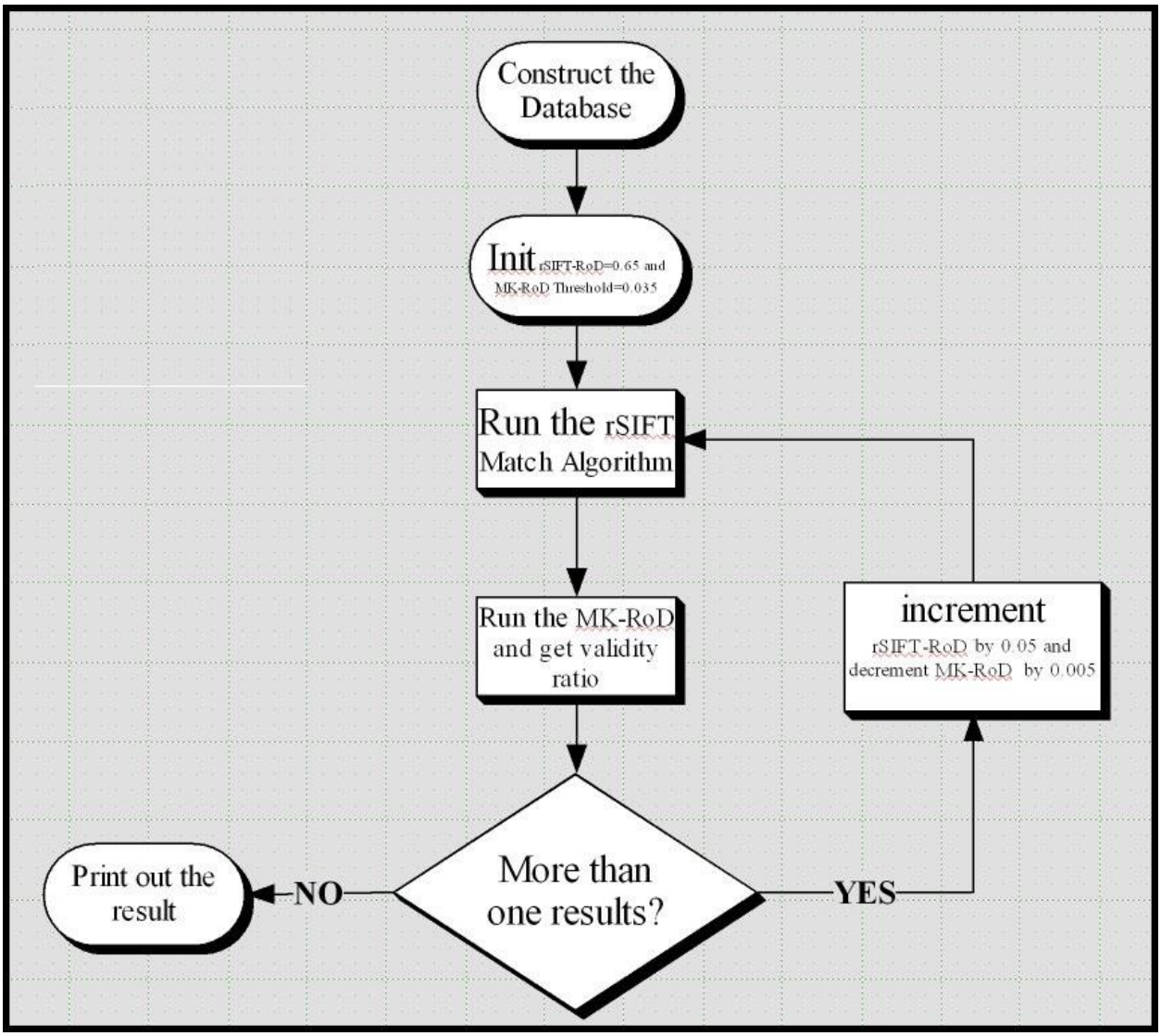
2.4.3. Edge Based Features Histogram of Oriented gradients: This concept was introduced by Dallal and Triggs in 2005 primarily for object detection. The gradient angle and magnitude is measured and binned in a histogram. Localization of the features is a very important aspect in HOG . Hand shape recognition using Distance transform and shape decomposition: In the work proposed by Junyeong Choiet al the possible hand poses or hand shapes are stored in the database. The input hand pose is compared with the database using distance transform of the image and histogram of the image . Moreover the hand regions are decomposed into fingers and palm regions and compared with the decomposed fingers and palm region stored in the database.

Disadvantages of existing techniques:

The disadvantage of using single cameras is that if the quality of the camera is too low then the detail that can be extracted from such a camera is very much lesser. Also the camera must be able to compensate to a certain extent to the illumination variance of the environment. Thus a balance between the cost and efficiency of the device used to take in input. . The down side of using multiple cameras is that the cost multiplies. Also since multiple images have to be processed the execution time increases.

PROPOSED WORK:

By incorporating all the above mentioned algorithms, we have developed a system which successfully is able to recognize the hand patterns when the images are taken as input as well when the image is captured using live motion capture.



Following are the codes used:

Appending the images:

```
% im = appendimages(image1, image2)
%
% Return a new image that appends the two images side-by-side.

function im = appendimages(image1, image2)
```

```
% Select the image with the fewest rows and fill in enough empty rows
%   to make it the same height as the other image.
rows1 = size(image1,1);
rows2 = size(image2,1);

if (rows1 < rows2)
     image1(rows2,1) = 0;
else
     image2(rows1,1) = 0;
end

% Now append both images side-by-side.
im = [image1 image2];
```

This function returns the locations of the most 'depth' amount of maximum values and stores the location information inside the 'Selecteds'

```
function Selecteds=findMax(results,depth);
Selecteds=zeros(1,26);
temp=1;

for j=1:depth
    if(temp~=0)
        [temp location]=max(results);
        if(temp~=0)
            results(location)=0;
            Selecteds(location)=location;
        end
    end
end



input is the input(query) image
% distRatio is the distance ratio parameter of SIFT
% threshold is the threshold value for the MK-RoD
% Selecteds specifies the selected database image for the process
for i=1:size(Selecteds,2)
    if(i==Selecteds(i))
        % Match the images and get the matched keypoints' data
        [match1,match2,cx1,cy1,cx2,cy2,num]  = match(dataBase(Selecteds(i),:),
input, distRatio);

        % If the number of Matched Keypoints are greater than 2, start the
        % algorithm
        if num>2
            %Calculate the distances of the matched keypoints to the center of
the
            %keypoints
            for j=1:num
                distance1(j)=sqrt((match1(j,2)-cx1).^2+(match1(j,1)-cy1).^2);
                distance2(j)=sqrt((match2(j,2)-cx2).^2+(match2(j,1)-cy2).^2);
            end
```

```
            % Sum the distances and calculate the Distance Ratio Array.
            distanceSum1=sum(distance1);
            distanceSum2=sum(distance2);

            if(distanceSum1==0)
                distanceSum1=1;
            end

            if(distanceSum2==0)
                distanceSum2=1;
            end

            for j=1:num
                distanceRatio1(j)=distance1(j)./distanceSum1;
                distanceRatio2(j)=distance2(j)./distanceSum2;
            end

            % Mask the distances by taking the absolute which are below the
            % algorithm's threshold.
            % This operation is done in order to determine the similar
            % pattern of 'the matched keypoints+ the center of the matched
            % keypoints'.
            % The absolute of the difference of the points which are below
            % the given threshold are treated as valid matched keypoint.
            distanceMask=abs(distanceRatio1-distanceRatio2)<threshold;

            %Calculate the total valid points by summing the distanceMask
            distanceMaskSum=sum(distanceMask);
        else
            % If the number of matched keypoints are not greater than 2
            % than the number of valid points are directly 0.
            distanceMaskSum=0;
        end
        % Store the results
        results(i,1)=cx1; % X position of the database image's center point
        results(i,2)=cy1; % Y position of the database image's center point
        results(i,3)=cx2; % X position of the input(query) image's center point
        results(i,4)=cy2; % Y position of the input(query) image's center point
        results(i,5)=num; % Number of matched keypoints
        results(i,6)=distanceMaskSum; % Number of valid matched keypoints

        % Calculate the validity ratio of the keypoints simply by dividing
        % the valid matched keypoints by the matched keypoints.
        % Store the ratio inside the results array.
        if(num==0);
            validRatio=0;
        else
            validRatio=distanceMaskSum/num;
        end
        results(i,7)=validRatio;
        results(i,8)=i;

% Reset the parameters for the next iteration
        distanceRatio1=0;
        distanceRatio2=0;
        distanceMask=0;
        distanceMaskSum=0;
```

```
    else
        results(i,:)=0;
    end
end




function results = hgr(input);

load theHGRDatabase
%The locations of database images are stored in "dataBase(i,:)"
% For accessing the path of 'B' character use dataBase(2,:).
% Note that 2 denotes  the 2nd character in the alphabet(i.e. 'B').
% Step1) Initialize the critical parameters
distRatio=0.65; %Distance Ratio for the SIFT Match methods. % default distRatio
= 0.65
threshold=0.035; % default threshold=0.035
distRatioIncrement=0.05; % default distRatioIncrement=0.05
thresholdDecrement=0.005; % default thresholdDecrement=0.005

% 'Selecteds' indicate the selected Database Images.
% If you add 1.jpg to the Database folder, you have to change the first
% number as 1. For our case, you have to make the first 0 of 'Selected'as 1.
% Also note that, this array stores the candidate database images.
% At the end of the algorithm only 1 selected (matched) image is left
% inside this array
Selecteds=[0 2 3 0 0 0 0 8 9 0 0 12 0 0 15 0 0 0 0 0 0 0 0 0 25 0];

% 'mask' denotes the maximum character number (The maximum charachter
% number for ASL (American Sign Language) is set as 26.
mask=26;

% 'check' specifies the number of checks done so far.
% it is set as 1 for the initial check/test.
check=1;
% Since there isn't any process, the initial value of the matchFound is 0.
matchFound=0;

% StringArray is used for outputing the equivalent ASCII character
StringArray=Selecteds;
StringArray=StringArray+64;
% ---------------------------------------------------------------
% Step2) Run the main algorithm
while(sum(mask)>1)
    % Investigate the function 'formResults' for the details of the
    % algorithm
    results=formResults(input,distRatio,threshold,Selecteds);

    if(mask<=2)
        % Select the best candidate
        Selecteds=findMax(results(:,7),1); %Note that 7th field of results hold
the validity ratio of the validly matched keypoints
    else
        % Select the 3 best candidates
```

```
        % Note: This section could be changed by replacing the 'depth' of 3 as
        % the numberOfBestCandidates by setting a separate ValidityRatio
threshold after the first iteration.
        % For demonstration purposes it is left as 3.
        Selecteds=findMax(results(:,7),3);
    end

    % Inform that the n'th check is done and increment the check for the
    % next possible iteration
    disp('---------------');
    fprintf('Check %d Done.\n',check);
    disp('---------------');
    check=check+1;

    % Increment the distRatio for the next iteration for the selected
    % candidate database images in order to find more matched keypoints
    distRatio=distRatio+distRatioIncrement;
    if(distRatio>=0.9)
        distRatio=0.9;
    end

    % Decrement the threshold value in order to find more valid keypoints.
    threshold=threshold-thresholdDecrement;
    if(threshold<=0.01)
        threshold=0.01;
    end
    % Store the Selecteds inside 'mask' and enforce it to be under 1 in
    % order to exit the loop if only 1 selected item left. In the case of
    % having 2 candidates the loop will continue.
    mask=(Selecteds)./(Selecteds+1);
end
fprintf('End of tests...\n');
% -----------------------------------------------------------------
% Step3) Printing and displaying the result
for i=1:26
    if (Selecteds(i)~=0)
        matchFound=1;
        inputImage=imread(input);
        outputImage=imread(dataBase(Selecteds(i),:));
        imshow(appendimages(outputImage,inputImage));
        title('Matched Database Image -versus- Input Image');
        fprintf('Match Found: %c char.\n',StringArray(i));
    end
end

if(matchFound==0)
    fprintf('No match found...\n');
end



initialize the dataset %after the prototype make it .mat file for
speeding up the process
for i=1:26
    if(i<10)
        dataBase(i,:)=['Images/Database/' int2str(i) '.jpg '];
    end
```

```
    if(i>9)
        dataBase(i,:)=['Images/Database/' int2str(i) '.jpg'];
    end
end

save theHGRDatabase



wid = sprintf('Images:%s:obsoleteFunction',mfilename);
str1= sprintf('%s is obsolete and may be removed in the future.',mfilename);
str2 = 'See product release notes for more information.';
%warning(wid,'%s\n%s',str1,str2);

y = size(x,3)==3;
if y
   if isa(x, 'logical')
      y = false;
   elseif isa(x, 'double')
      % At first just test a small chunk to get a possible quick negative
      m = size(x,1);
      n = size(x,2);
      chunk = x(1:min(m,10),1:min(n,10),:);
      y = (min(chunk(:))>=0 && max(chunk(:))<=1);
      % If the chunk is an RGB image, test the whole image
      if y
         y = (min(x(:))>=0 && max(x(:))<=1);
      end
   end
end




% For each descriptor in the first image, select its match to second image.
des2t = des2';                           % Precompute matrix transpose
for i = 1 : size(des1,1)
   dotprods = des1(i,:) * des2t;         % Computes vector of dot products
   [vals,indx] = sort(acos(dotprods));  % Take inverse cosine and sort results

   % Check if nearest neighbor has angle less than distRatio times 2nd.
   if (vals(1) < distRatio * vals(2))
      myMatch(i) = indx(1);
   else
      myMatch(i) = 0;
   end
end



% show the matched points for image-1
%figure('Position', [100 100 size(im1,2) size(im1,1)]);
%imagesc(imread(image1));
%hold on;
j=1;
for i = 1: size(des1,1)
  if (myMatch(i) > 0)
```

```
    %plot(loc1(i,2),loc1(i,1),'ro');
    match1(j,1)=loc1(i,1);
    match1(j,2)=loc1(i,2);
    match2(j,1)=loc2(myMatch(i),1);
    match2(j,2)=loc2(myMatch(i),2);
    j=j+1;
  end
end
%title('Matched Points for "Image-1"');
%hold off;

% show the matched points for image-2
%figure('Position', [100 100 size(im2,2) size(im2,1)]);
%imagesc(imread(image2));
%hold on;
%for i = 1: size(des1,1)
%  if (myMatch(i) > 0)
%    plot(loc2(myMatch(i),2),loc2(myMatch(i),1),'go');
%  end
%end
%title('Matched Points for "Image-2"');
%hold off;

% Create a new image showing the two images side by side.
im3 = appendimages(im1,im2);

% Calculate the center points
num = sum(myMatch > 0); %matched points

if (num > 1)
    s1=sum(match1);
    s2=sum(match2);
    cy1=s1(1)/num;
    cx1=s1(2)/num; %Center point for the database image

    cy2=s2(1)/num;
    cx2=s2(2)/num; %Center point for the input image
end

% % Show matched points for "Image-1" and "Image-2"
% figure('Position', [100 100 size(im3,2) size(im3,1)]);
% imagesc(im3);
% hold on;
% cols1 = size(im1,2);
% for i = 1: size(des1,1)
%   if (myMatch(i) > 0)
%     plot(loc1(i,2),loc1(i,1),'ro');
%     plot(loc2(myMatch(i),2)+cols1,loc2(myMatch(i),1),'go');
%   end
% end
%
% if (num > 1)
%     plot(cx1,cy1,'bx');
%     plot(cx2+cols1,cy2,'bx');
% end
%
% title('Matched Points for "Image-1" and "Image-2"');
```

```
% hold off;

fprintf('Found %d matches.\n', num);
disp('---------------------------------');
```

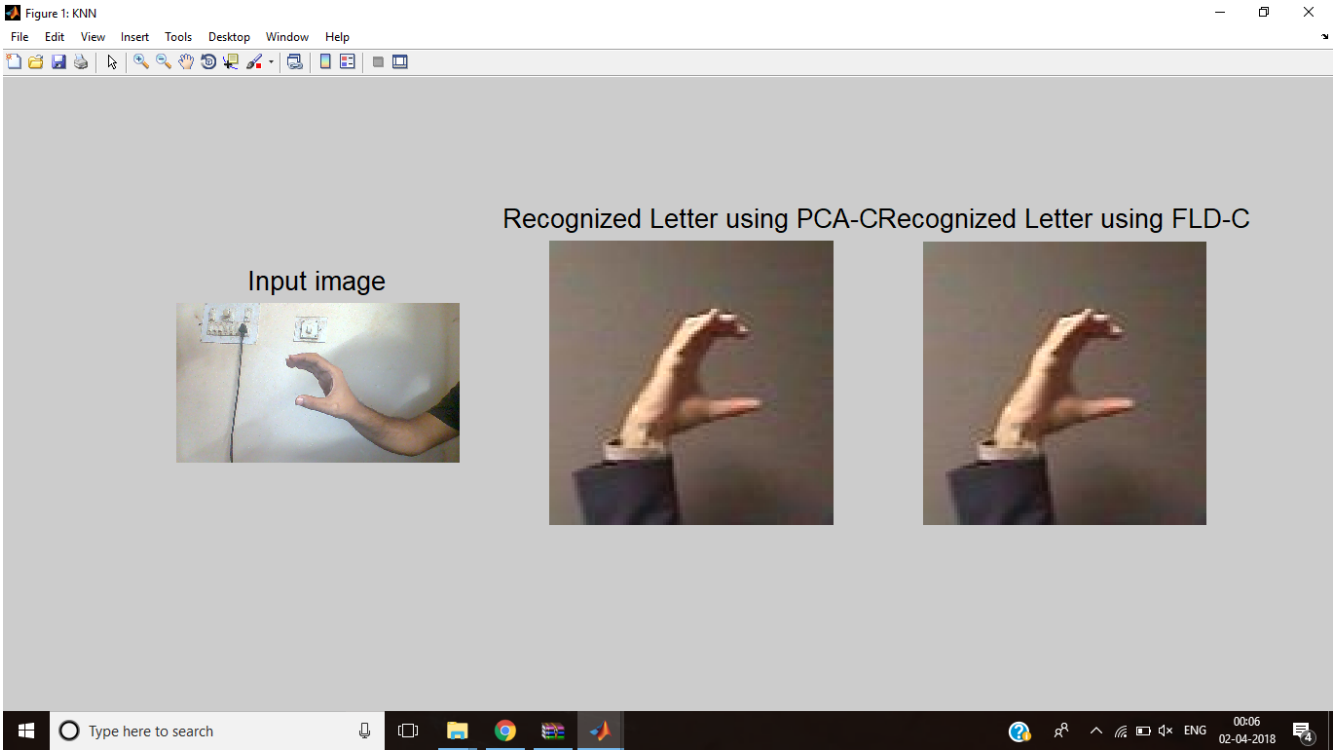
SIFT

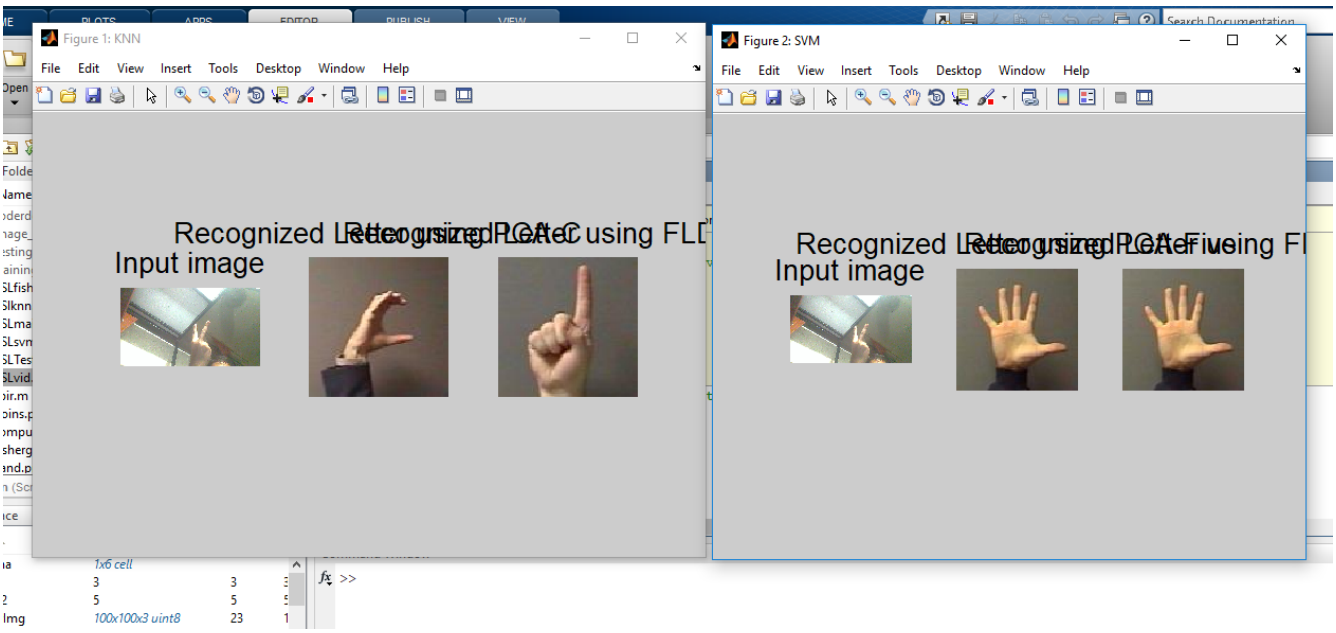
```
f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

% Call keypoints executable
if isunix
    command = '!./sift ';
else
    command = '!siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

% Open tmp.key and check its header
g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end

% Creates the two output matrices (use known size for efficiency)
locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

% Parse tmp.key
for i = 1:num
    [vector, count] = fscanf(g, '%f %f %f %f', [1 4]); %row col scale ori
    if count ~= 4
        error('Invalid keypoint file format');
    end
    locs(i, :) = vector(1, :);

    [descrip, count] = fscanf(g, '%d', [1 len]);
    if (count ~= 128)
        error('Invalid keypoint file value.');
    end
```

```
    % Normalize each input vector to unit length
    descrip = descrip / sqrt(sum(descrip.^2));
    descriptors(i, :) = descrip(1, :);
end
fclose(g);

%eval('!rm -f tmp.pgm');
%eval('!rm -f tmp.key');
```

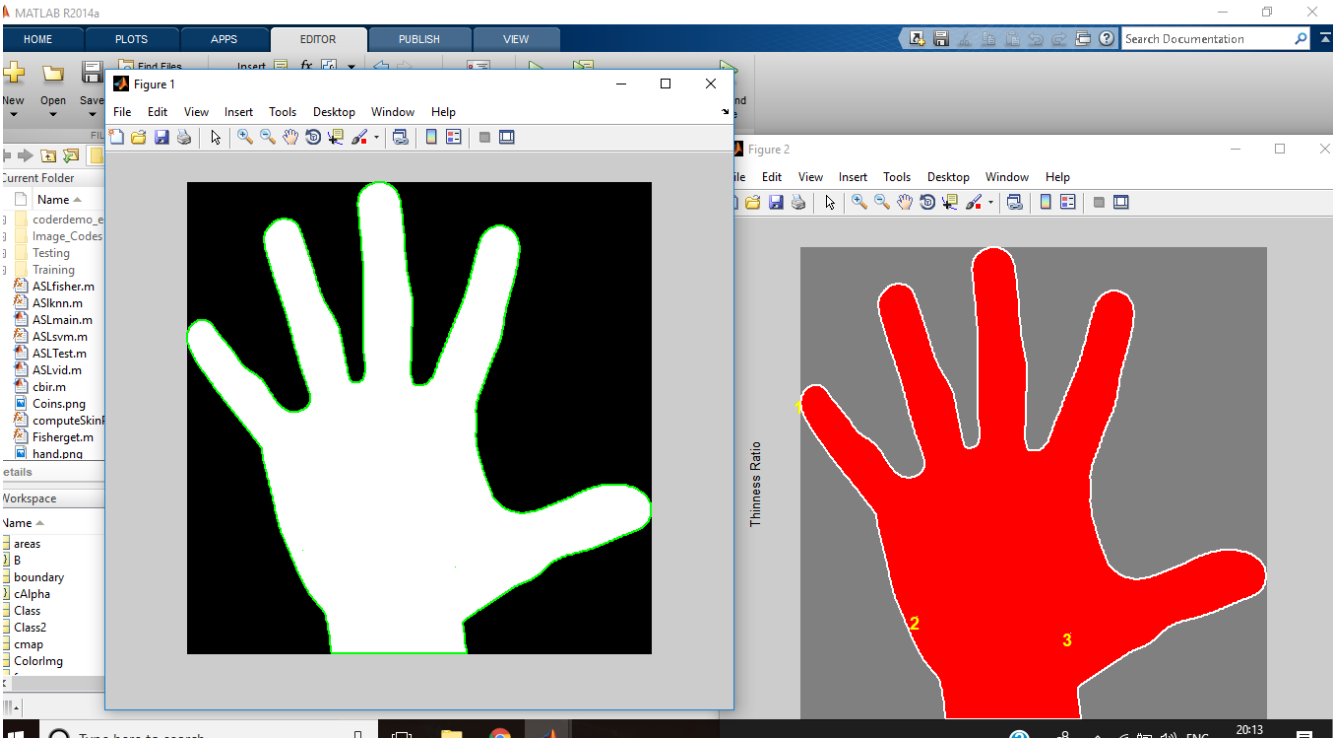
PERFORMANCE ANALYSIS:

We get the following output after running the codes:





Edge Detection:



CONCLUSION :

We are successfully able to execute our proposed system and identify all the different alphabets of the Sign Language.