**Chapter 1**

**INTRODUCTION**

* 1. **Introduction**

Digital cameras are now integrated into personal computers, mobile cellular devices and handheld computers. These devices usually include a powerful microprocessor, capable of performing millions of computations per second. Using these resources it is now possible to provide an efficient Human-computer interface which will capture the gestures of the disabled and classify it based on the predefined gesture-interpretation definitions.

As a special case of human-computer interaction, human robot interaction is imposed by several constraints the background is complex and dynamic; the lighting condition is variable; the shape of the human hand is deformable; the implementation is required to be executed in real time and the system is expected to be user and device independent.

This project aims to provide an inexpensive but an efficient system that will capture the hand gestures on real time basis, recognize these gestures after removing the background noise present and finally return the textual/voice description of the matched gesture accurately.

* + 1. **Gestures**

A gesture is a form ofnon-verbal communication made with a part of the body, used instead of or in combination with verbal communication. The language of gesture allows individuals to express a variety of feelings and thoughts, from contempt and hostility to approval and affection. Most people use gestures and body language in addition to words when they speak. The use of gesture as language by some ethnic groups is more common than in others, and the amount of such gesturing that is considered culturally acceptable varies from one location to the next.

**Types of Gestures**

Gestures are mostly culture-specific. Pointing gesture is probably one of the exceptions. Some gestures closely coordinated with speech. Gestures are mainly of two types. Hand & arm gestures and head gestures.

* **Hand & arm gestures**

Pointing Gestures

Sign Language

Hello, Thumbs up/down, victory sign, the finger, call me

* **Head gestures**

Nodding, head shaking, turning, pointing

Body gestures

Don’t know / shrug

* + 1. **Hand Gestures**

A line diagram showing the different hand gestures are shown below.

Communicative

(hand gesture)

e

Direct Manipulation

Spontaneous

(Natural)

Language-like

gesture

Pantomimes

Emblem

**SIGN LANGUAGE**

**Hand**

Unintentional gesture

Intentional gesture

**Figure 1.1: Hand Gesture chart.**



**Figure1.2: Different hand gestures.**

* + 1. **Gesture Recognition**

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Current focuses in the field include emotion recognition from the face and hand gesture recognition.

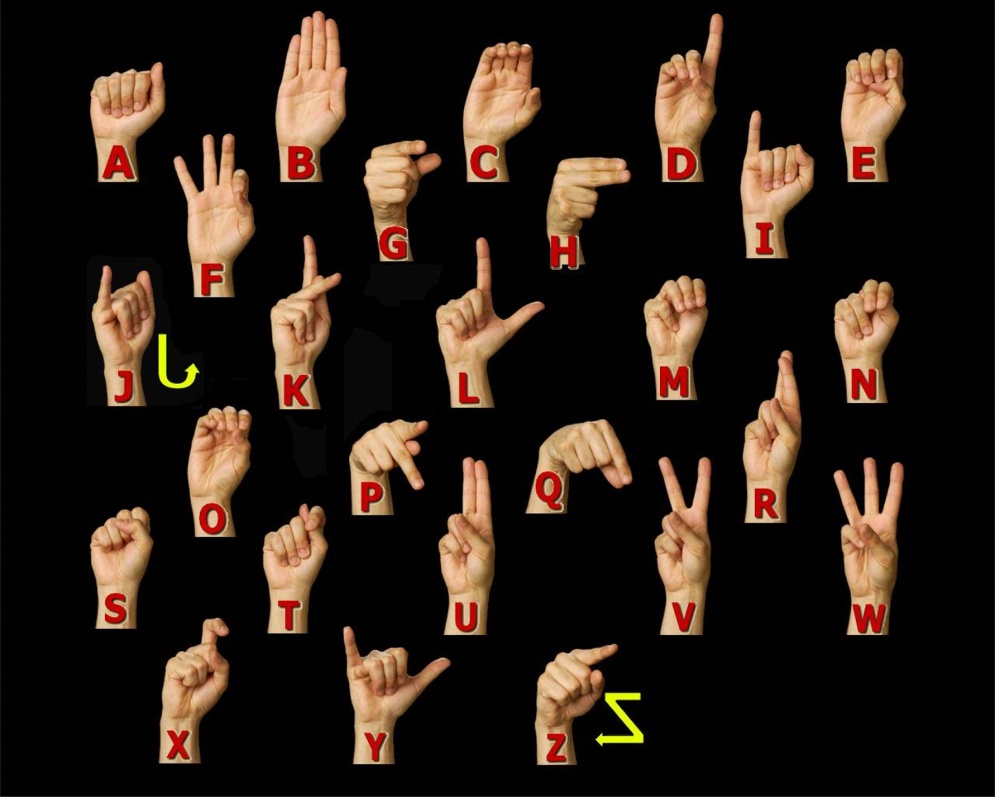
Many approaches have been made using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of posture, gait, [proxemics](http://en.wikipedia.org/wiki/Proxemics), and human behaviors is also the subject of gesture recognition techniques.

Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse. Gesture recognition enables humans to interface with the machine (HMI) and interact naturally without any mechanical devices.

* 1. **Motivation**

Communication is the activity of conveyinginformation through the exchange of thoughts, messages, or information, as by speech, visuals, signals, writing, or behavior. Majorly, speech is the type of communication that we humans prefer. Unfortunately not every human is blessed with this ability. Due to this disability people from deaf/dumb community can’t express their thoughts or even if they do it can’t be interpreted by other humans.

Individuals belonging to deaf and dumb community across the globe communicate with each other through a combination of hand gestures and facial expressions, from which hand gestures form a significant part. American Sign Language (ASL) is one of the standards for communication in English speaking communities which provides a template of signs consisting of alphabets (A-Z), digits (0-9) and classifiers (nouns).



**Figure 1.3: Introduction to ASL.**

This system aims at removing the communication barrier between normal people and deaf and dumb community which arises due to inability of normal people to understand American Sign Language.

* 1. **Problem Statement**

Gesture recognition systems are intended to remove the communication barrier between the disabled by providing and efficient, fast and effortless operation of the system. Gestures with a variety of size, shape and colors should be detected. No external tool usage will be flexible. The system should be working under any background having noise in terms of irregular backgrounds, variety of colors and voice. This entire if done on the real time basis will be beneficial.

Current systems work effectively if the user is wearing an external glove with sensors in it and which is connected to the system computers via Bluetooth. Additionally the user is expected to keep the hands in proper alignment of the camera so that the gesture is captured with correct angle or extra cameras are needed to capture gestures from every angle and select the best image to be processed out of it. Most of the systems require a user to record a video of some length which is provided as an input to the system which ultimately slows down the processing speed.

Without an effective system that does not consider the maximum requirements would only lead to broadening the communication gap and increasing frustration among the deaf/dumb communities.

Sign Language Recognition system (SLRS) intends to implement the real time gesture recognition system with re-orientation and depth analysis. SLRS will explore a series of image processing techniques such as thresholding, segmentation, background subtraction and gap filling to remove the irregularities in the background environment of user and the captured image.

**Chapter 2**

**LITERATURE REVIEW**

To further improve our knowledge on the topic and the possible ways to implement the system we have referred various IEEE papers and white papers. The research done has helped us grasp the basics of Real Time Gesture Recognition Systems. Through the articles reviewed an overview of the system along with the various advantages and the disadvantages of the various systems have been obtained. This helped us to select the appropriate approach and model for the report. The algorithms provided in these papers have provided an idea about the implementation phase.

**2.1 Related Work**

**Hand gesture recognition algorithm based on grayscale histogram of the image.(2010)**

From this paper we understood the steps involved in a basic gesture recognition system.The pre-processing operations such as gray scale conversion, thresholding and edge detection were included in our system by referring this paper. Furthermore we came to know the disadvantages in gesture recognition by histogram plotting[1].

**Real-time Sign Language Recognition based on Neural Network Architecture.(2011).**

The concept of operating a gesture recognition system under a continuous video frame was referred from this paper. Capturing of individual frames from a video and processing it was referred from this paper, additionally classification of gestures based on neural networks was studied[2].

**Hand Gesture Recognition in Images and Video**

Alternative methods of the steps in a gesture recognition system were studied from this paper[3].

# A background subtraction method using color information in the frame averaging process

# A gesture recognition system needs only the gesture part of the hand and all other background portion is considered as unwanted background and hence should be eradicated for accurate matching of gestures. This is portion of removal of background or background subtraction was referred from this paper[4].

**An adaptive real-time skin detector based on Hue thresholding: A comparison on two motion tracking methods.**

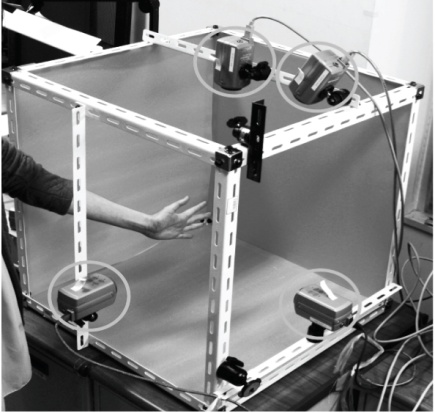
Detection of skin color is easier from HSV color model. From this paper we studied the algorithm of skin detection algorithm based on adaptive Hue thresholding and its evaluation using two motion detection techniques[5].

**2.2 Existing system**

Hand gestures represent a powerful non-verbal context-dependent human communication modality. The expressiveness of hand gestures can be explored to achieve natural human-computer interactions in a smart habitat environment. We discuss vision-based hand tracking and gesture classification, focusing on tracking the bare hand and recognizing hand gestures without the help of any markers and gloves. The existing hand gesture recognition system use either vision based or sensor based solution.

**Vision Based Systems**

* Vision-based solutions collect human motion from one or multiple cameras.
* Vision-based devices can handle properties such as texture and color for analyzing a gesture, while sensor cannot.
* A tracker also needs to handle changing shapes and sizes of the gesture-generating object (that varies between individuals), other moving objects in the background, and noise.
* Vision-based techniques can vary among themselves in 6 ways.
  1. The number of cameras used.
  2. Their speed and latency.
  3. The structure of environment (restrictions such as lighting or speed of movement).
  4. Any user requirements (whether user must wear anything special).
  5. The low-level features used (edges, regions, silhouettes, moments, histograms).
  6. Whether 2-D or 3-D representation is used.

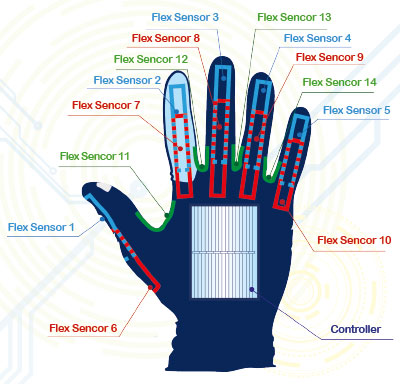
****

**Figure 2.1: Example of vision based system**

**Sensor based systems**

Sense board is a U-shaped device that fits around the hand parallel to the wrist. It contains three accelerometers to detect movement of the hand and two rotational sensors which detects wrist and elbow rotation. This data is sampled at regular intervals of 25Hz and sent to a computer using Bluetooth.

The data transmitted is a raw stream of sensor outputs which has not been filtered in any way. The task of the gesture recognition software is to attempt to classify this input, and output the most probable gesture to the robot control software.



**Figure 2.2: Sensor hand gloves.**

Sensor-based solution typically requires the user to wear a cumbersome device nowadays. This hinders the ease and naturalness of the user’s interaction with the computer. According to Moor’s law, sensors are getting smaller and cheaper as time goes on. It will be a pervasive in future, we believe.

Researchers from UC Berkeley developed a wearable motion sensor network to recognize human activities. They introduced distributed pattern recognition. Each mote (mote is a sensor network terminology means node interchangeably) can do classification locally and sends events to global classifier to recognize motion. Compare to centralized approach, it dramatically reduced the communication and energy consumption. It increases the robustness of system, user can active necessary node when needed on the fly. HMM is used to recognize gesture. They manually do the segmentation on data stream.

The interesting point is a procedure based on adding noise-distorted signal duplicates to training set is applied and it is shown to increase the recognition accuracy while decreasing user effort in training. With the help of noisy data, user can have only 2 training samples for each gesture.



**Figure 2.3:Sensor based systems.**

**Drawbacks of Existing System**

**Vision based**

1. Vision base system needs many cameras to capture a moment. It required more processing speed.
2. High cost as more than one camera required.

**Sensor based**

1. Sensor based system required an external gloves (extra cost involved in system).
2. High maintenance.

To remove these drawbacks and make system more efficient proposed system is developed.

**Chapter 3**

**PROPOSED SYSTEM**

**3.1 Introduction**

Sign language recognition system is a research based project which intends to capture process and recognize hand gestures through a video source at real time. This system will take a video stream as an input, convert them into image frames and then by applying image processing techniques remove the background noise.

Later perform background subtraction i.e. make the background totally black and only keeping the hand of the user as white. This is done by selecting an appropriate threshold or cutoff value.

The processed images are than given as input to the classification stage where the images will be matched to the pre-recorded standard gestures which are stored in the database along with their textual representation. The gestures having the maximum resemblance to the captured image will be selected and its associated text will be displayed as an output to the user.

Our Project is mainly divided into three modules namely:

1. Capturing the hand gestures from a video source:

In this phase a live video is recorded from a digital camera, converted into image frames and simultaneously passed to the next stage for further processing.

1. Pre-processing, extraction and orientation.

Image frames from the first phase are given as input to this phase where the images are passed through a set of image processing techniques like edge detection, segmentation, and thresholding, gray level slicing and smoothing. Gap filling is done on the images if certain parts of the images are left undetected.

Relevant images are then re-oriented to the zero degree anglei.e. they are aligned to proper positions.

Resultant images are resized to the standard image size as saved template image so that the process gets simplified.

1. Image - to - gesture mapping:

The processed images are provided to these phase where the images are compared to the pre-recorded set of gestures and the ones having highest probability will be selected and associated output displayed on screen.

**3.2 Methodology**

Sign language recognition system is a real time gesture recognition system, which starts by processing a live video. This system has used an OpenCV library which provides a set of methods for image processing and computer vision operations.

The proposed system is working in a real time environment, which is done by executing the source file in a continuous for loop and performing all the operations in that particular loop. For simpler processing in system, set the camera input frame width and height to a standard size of 640 X 480, which is also the size of the template images that are matched with the preprocessed input images.

Initially a video is divided into individual frames by extracting an individual image from it with the help of a method named cvQueryFrame(), which takes instance of a camera as its input. This method returns an image as its output anytime it is called. The output image is the one which will be passed through a set of preprocessing operations.

Sequence of preprocessing operations performed on the image extracted from the camera as well as the template images are as follows.

1. Convert both the images from BGR to Grayscale format, using **cvCvtColor()** method.
2. Perform smoothing operation by applying a **Gaussian filter**, which is a low pass filter to reduce image noise and reduce detail.
3. On the smooth image thresholding operation is applied on a grayscale image to convert it into binary image using **cvThreshold()** function, system uses 100 as a threshold value i.e. image pixels under 100 will be converted to 0, and over 100 is converted to 1.
4. Apply canny edge detection operation on the binary image to detect the discontinuities in the image in terms of image brightness. For this **cvCanny(input,output,threshold1,threshold2,3)**functionis used . The function finds the edges on the input image image and marks them in the output image edges using the canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking; the largest value is used to find the initial segments of strong edges.
5. Finally apply the two morphological operations named erosion and dilation on the edge detected output images. The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image. The basic effect of erosion on a binary image is to erode away the boundaries of regions of foreground [pixels](http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixel.htm) (i.e. white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger. **cvErode()** and **cvDilate()** are the two methods that we used with structuring element of size 3X3 with number of iterations as 5.

**Input Image**

**(RGB)**

**Gray scaled Image**

**cvCvtColor()**

**Output Image**

**(After applying smoothing and thresholding)**

**Gaussian filter and cvThreshold ().**

**Edged image (0nly outline of gesture)**

**cvCanny().**

**Noise free Image**

**cvErode() and cvDilate().**

**Stop**

**Start**

**Figure 3.1: Preprocessing Stages.**

These are the five preprocessing operations applied on both, the template image and input frame.

After processing the images, find the contours using **cvFindContours** method. Contour lines are curved or straight lines on a map describing the intersection of a real or hypothetical surface with one or more horizontal planes.

The operations performed until now were the part of gesture detection, the following part describes about gesture recognition.

For gesture recognition system, use **cvMatchShapes()** and distance transform technique combined, which increased the accuracy of the gestures detected.

1. **cvMatchShapes()** method calculates the Hue moments of the two objects provided,  an **image moment** is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments, usually chosen to have some attractive property or interpretation. Image moments are useful to describe objects after segmentation.which are found via image moments include area (or total intensity), its [centroid](http://en.wikipedia.org/wiki/Centroid), and [information about its orientation](http://en.wikipedia.org/wiki/Image_moments#Examples_2).
2. Secondly we developed a function named **computeDistanceTransform ()** which itself applies a series of steps before applying distance transformation on the image.

Following are the steps in the method :-

* Initially we threshold the image using a value of 50 to convert it into binary form.
* Distance transform of the threshold image is calculated by using **cvDistTransform ()** function. The functions cvdistTransform calculate the approximate or precise distance from every binary image pixel to the nearest zero pixel. (For zero image pixels the distance will obviously be zero). In this we use a mask of 5X5 with distance type as **CV\_DIST\_L2.**
* Perform scaling and shifting of the pixels after applying the distance transform for orienting image back to its previous position which shifts due to distance transformation.
* Finally scale the image with the factor of 1 and return it as an output.

**Start**

**Input Image**

**(Any edged image).**

**Apply Distance transform**

**Perform scaling and shifting.**

**Stop**

**Figure 3.2: Distance transform.**

Then in syatm calculate the total resolution of the image returned by the **computeDistanceTransform(),** then for every pixel in the image difference between the template image and the input image pixel is calculated. If the difference value is less than the threshold value, 35 in our case, the counter is incremented.

**//**method developed for calculating distance transform.

**IplImage\* computeDistTransform(int flag)**

**{**

**IplImage\* dist = 0;**

**IplImage\* dist8u1 = 0;**

**IplImage\* dist8u2 = 0;**

**IplImage\* dist8u = 0;**

**IplImage\* dist32s = 0;**

**IplImage\* edge = 0;**

**IplImage\* gray = 0;**

**if(flag == 1)**

**{**

**gray = cvCloneImage(ginput);**

**edge = cvCloneImage(smoothginput);**

**}**

**else**

**{**

**gray = cvCloneImage(gsampleinput);**

**edge = cvCloneImage(smoothgsampleinput);**

**}**

**dist8u1 = cvCloneImage( gray );**

**dist8u2 = cvCloneImage( gray );**

**dist8u = cvCreateImage(cvGetSize(gray), IPL\_DEPTH\_8U, 3 );**

**dist32s = cvCreateImage(cvGetSize(gray), IPL\_DEPTH\_32S, 1 );**

**dist = cvCreateImage( cvGetSize(sampleInput), IPL\_DEPTH\_32F, 1 );**

**intedge\_thresh = 50;**

**intmsize = mask\_size;**

**int \_dist\_type = dist\_type;**

**cvThreshold( gray, edge, (float)edge\_thresh, (float)edge\_thresh, CV\_THRESH\_BINARY );**

**cvDistTransform( edge, dist, \_dist\_type, msize, NULL, NULL );**

**cvConvertScale(dist, dist, 150.0, 0 );**

**cvPow(dist, dist, 0.5 );**

**cvConvertScale(dist, dist32s, 1.0, 0.5 );**

**cvAndS( dist32s, cvScalarAll(255), dist32s, 0 );**

**cvConvertScale( dist32s, dist8u1, 1, 0 );**

**return dist8u1;**

**}**

**Ratio** of the image is now calculated by the formula :-

**intmaxcount = dt1->width \* dt1->height;**(dt1 is an image).

**ratio = (float)count/(float)maxcount;**

Now if the ratio is within the range of 0.5 and 0.2, additionally if the value of **cvMatchShapes()** is less than 0.4, it is declare that the input image is matched with the template image and display the appropriate text of the matched gesture.

**if ( (ratio >= 0.1 && ratio <= 0.9) && (compare >= 0.0 && compare <=0.5))**

**{**

**//Display code**

**}**

If the result the combined computation is false and no match is found, apply the orientation logic of this system.

In the orientation method if no match is found with 0 degree angle, the template images are rotated by 90 degree angle clockwise 4 times and perform the above operations again. Even now if don’t get the values of ratio and cvMatchShapes () as required the template image is changed and the operations explained above are performed again.

//Method developed for orientation

**IplImage \*rotateImage(constIplImage \*src, intangleDegrees)**

**{**

**IplImage \*imageRotated = cvCloneImage(src);**

**if(angleDegrees!=0)**

**{**

**CvMat\* rot\_mat = cvCreateMat(2,3,CV\_32FC1);**

**// Compute rotation matrix**

**CvPoint2D32f center = cvPoint2D32f(cvGetSize(imageRotated).width/2, cvGetSize(imageRotated).height/2 );**

**cv2DRotationMatrix( center, angleDegrees, 1, rot\_mat );**

**// Do the transformation**

**cvWarpAffine(src, imageRotated, rot\_mat );**

**}**

**returnimageRotated;**

**}**

**Chapter 4**

**ANALYSIS**

**4.1 Requirement Analysis**

**4.1.1 Software Requirement**

Operating system: Windows xp,vista,7  
Front End : OpenCv 2.2 with Visual Studio 2010  
Integrated language : C++

**4.1.2 Hardware Requirement**

External device: camera  
Processor: Pentium/AMD  
Speed: 233MHz  
Ram: 2GB  
Hard disk: 40GB

**4.2 Functional Requirement**

In [software engineering](http://en.wikipedia.org/wiki/Software_engineering), a **functional requirement** defines a function of a [software system](http://en.wikipedia.org/wiki/Software_system) or its component. A function is described as a set of inputs, the behavior, and outputs.

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in [use cases](http://en.wikipedia.org/wiki/Use_case).

Generally, functional requirements are expressed in the form **"system must do requirement".**

The plan for implementing functional requirements is detailed in the system design.

Functional requirements of this system:-

* Input should be a video.
* Output should be displayed on text format.
* System should work on real time basis.

**4.3 Non-Functional Requirement**

A **non-functional requirement** is a [requirement](http://en.wikipedia.org/wiki/Requirement) that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

Non-functional requirements are often called **qualities** of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

Non-functional requirements are **"system shall be requirement".** The plan for implementing non-functional requirements is detailed in the systemarchitecture.

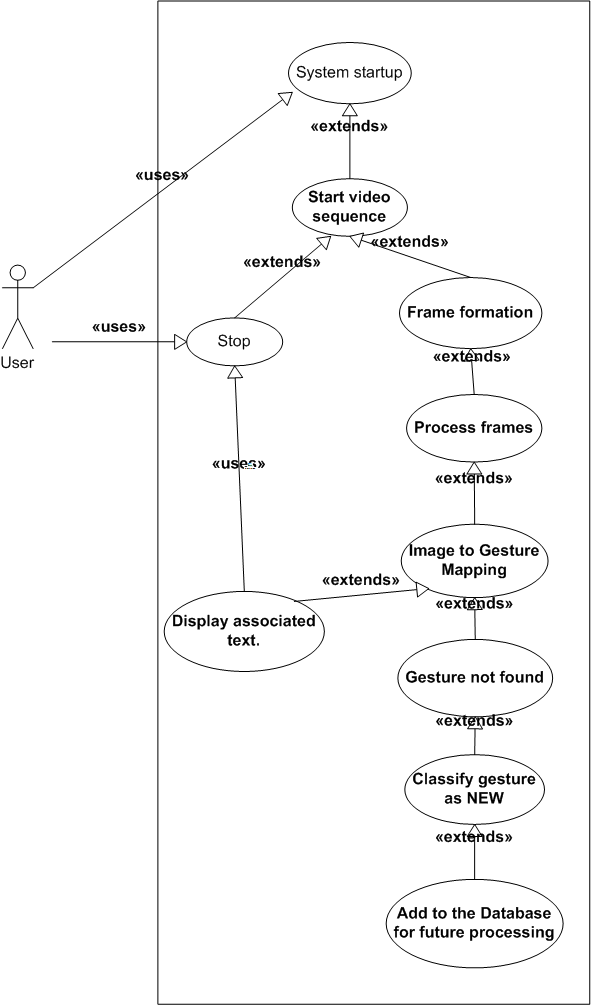
Non-Functional requirements of thissystem:-

* In any background and light condition it should give accurate result.
* Perform orientation on all angles. (In system considered only 4 angles).
* Output should be displayed within fraction of seconds.

**Chapter 5**

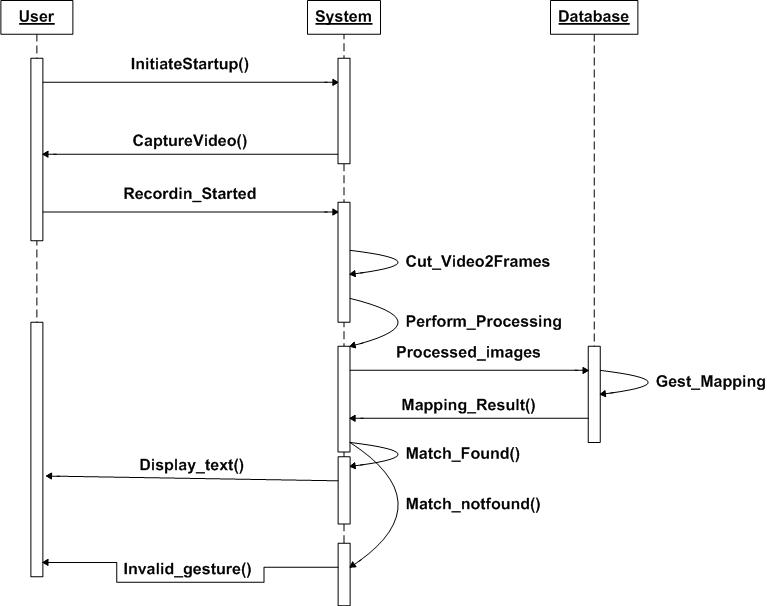
**DESIGN**

* 1. **Uml Diagrams**
     1. **Use Case diagram :-**

****

**Figure5.1: Use Case Diagram.**

* + 1. **Sequence Diagram:-**

****

**Figure 5.2: Sequence Diagram.**

* 1. **DATA FLOW DIAGRAM**

**Level 0**

**User**

**Image Folder**

**System**

**Show video**

**Show Output**

**Video capture**

**Perform matching**

**Return Matched Gesture**

**Video sequence**

**Pre-processed frames**

**Figure 5.3: Data Flow Diagram(level 0).**

**Level 1**

**User**

**SLRS**

**Image**

**Folder**

**Video Sequence capture**

**Show output**

**Apply Recognition techniques**

**Apply filters**

**Provide rotate method**

**Gesture Mapping**

**Return Text**

**Rotated Gesture**

**No. of images**

**Frames**

**Save gestures with plain background**

**Figure 5.4: Data Flow Diagram(level 1).**

**Chapter 6**

**SOFTWARE PLANNING**

**6.1 Timeline Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Task name** | **Jun** | **Jul** | **Aug** | **Sep** | **Oct** | **Nov** | **Dec** | **Jan** | **Feb** | **Mar** | **Apr** |
| 1. | Requirement Analysis/  Literature survey |  |  |  |  |  |  |  |  |  |  |  |
| 2. | Designing |  |  |  |  |  |  |  |  |  |  |  |
| 3. | Construction/Coding |  |  |  |  |  |  |  |  |  |  |  |
| 3.1 | Detection of gestures |  |  |  |  |  |  |  |  |  |  |  |
| 3.2 | Orientation analysis |  |  |  |  |  |  |  |  |  |  |  |
| 3.3 | Gesture mapping |  |  |  |  |  |  |  |  |  |  |  |
| 3.4 | Validation/Testing |  |  |  |  |  |  |  |  |  |  |  |

**Table 6.1: Ganttchart.**

**6.2 Task Distribution**

|  |  |
| --- | --- |
| **Name** | **Task performed** |
| Dhanashree Ganpule. | Requirement Analysis  Documentation  Gesture Refinement  Multiple gesture Recognition  Testing |
| Shaktidhar Dandapani. | Requirement Analysis  Designing  Gesture Detection  Orientation  Testing |
| Pratik Tamhankar. | Requirement Analysis  Documentation  Gesture detection and refinement  Gesture Mapping  Testing |

**Table 6.2: Task distribution.**

**Chapter 7**

**IMPLEMENTATION**

**7.1 Algorithm**

Step1: Start.

Step2: Take the video input from camera

Step3: Convert the video sequence into Frames.

Step4: Convert the each frame image and template image into gray scale image.

Step5: Create a concrete hand in gray scale (gap filling).

Step6: Detect the boundaries of the hand (Edge detection).

Step7: Perform distance transform on both images.

Step8:Orient the pre-processed image about vertical axis.

Step9: Resize the image according to database.

Step10: Pick the best match from database.

Step11: Show the equivalent text output on screen.

**System Flowchart**

**Convert the each frame image and template into gray scale image**

**B**

**Start**

**Take the video input from Came**

**Create a concrete hand in gray scale (Gap filling)**

**Convert the video sequence into Frames**

**Detect the boundaries of the hand(Edge detection)**

**A**

**Apply Low pass filter**

**Convert the gray image in binary format**

**Apply distance transform on images**

**Match Found**

**Stop**

**A**

**Perform matching of input frame and saved template image**

**Display the associated text**

**Rotated the template image**

**If**

**template rotated 4 times**

**Rotated template**

**Change template**

**Yes**

**Yes**

**No**

**No**

**Figure 7.1: System Flowchart.**

**Description of Flowchart**

System working starts by taking a video input from a camera, which contains a set of input gestures to be processed. Any video consists of a set of frames or images. To process every image, each individual frame is extracted. Every frame image along with the standard template images passes through a set of preprocessing operations. Preprocessing operations include grayscale conversion, low pass filtering, and grayscale to binary image conversion, edge detection and distance transformation respectively.

The intent of these preprocessing operations is to reduce the noise and extract the closest relevant features from the image. After preprocessing, matching between the input frame image and template image is done.

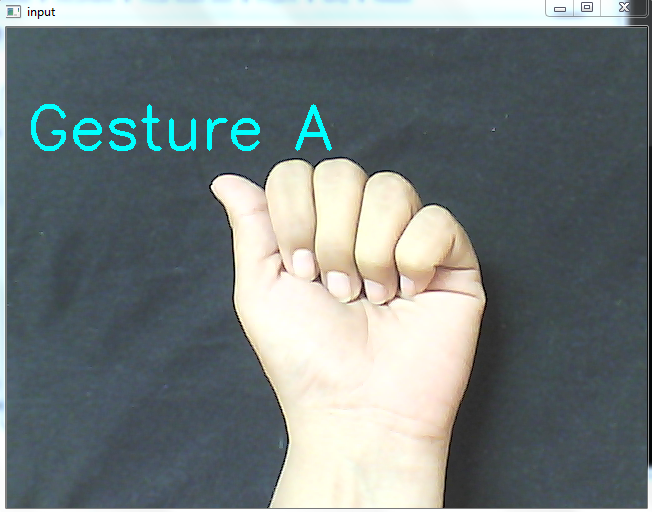
If matching is successful then the associated text is outputted and if further gestures are present then above operations are continued else the system stops.

But if matching is not successful then a orientation check is performed in which if the template image is rotated at least 4 times then the template image is changed with a new template. If not then the template image is rotated by 90 degree angle. In both cases the same pre-processing operations are repeated and the process continues until the system stops.

**7.2 Screen Shots**

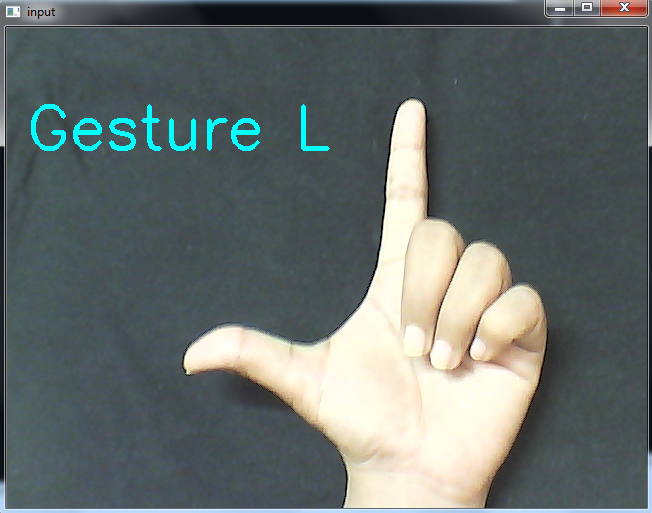
Recognized gestures with associated TEXT output:

**Alphabet A**



**Figure 7.2:Gesture A(plain backround).**

**Alphabet L**

****

**Figure 7.3:Gesture L(plain backround).**

**Alphabet I**



**Figure 7.4:Gesture I(plain backround).**

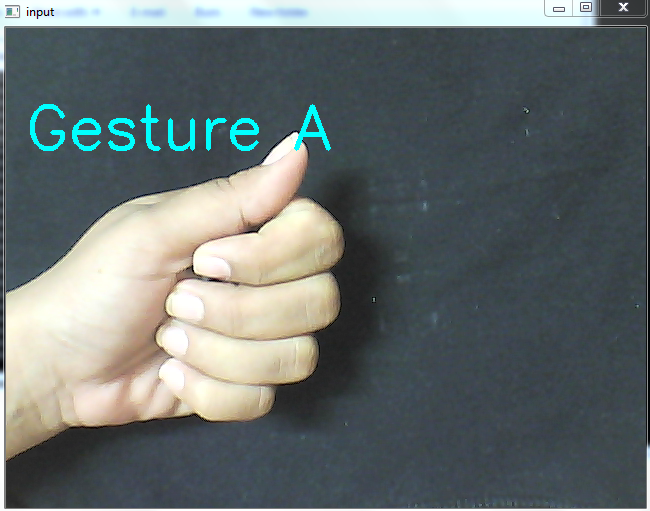
**Sign Hi**



**Figure 7.5:Gesture hi(plain backround).**

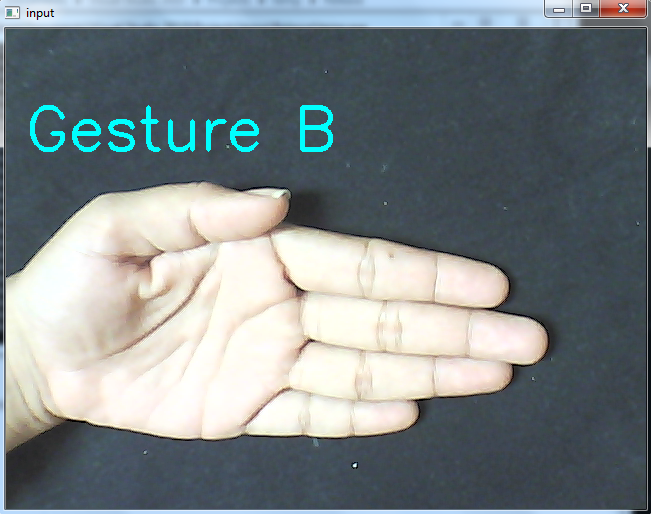
Output with ORIENTATION

**Alphabet A**



**Figure 7.6:Gesture A(with orientation).**

**Alphabet B**



**Figure 7.7:Gesture B(with orientation).**

**Alphabet I**



**Figure 7.8:Gesture I(with orientation).**

**Sign Hi**



**Figure 7.9:Gesture hi(with orientation).**

**7.3 Result Analysis**

Results: Output accuracy(based on 10 rounds of testing over plain and complex backgrounds with two levels of illumination.)

Percent calculation = (No.of hits/10)\*100.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Low**  **Illumination(LI)** | **Good Illumination(GI)** | **Detection Ratio(LI)** | **Detection Ratio(GI)** | **Average Detection** |
| **Plain Background** | Many faults but slight detection chances. | Good detection. | 30% | 80% | 55% |
| **Complex Background** | Almost no detection. | Moderate detection. | -- | 4% | 20% |

**Table 7.1: Gesture detection Analysis**

Considering orientation in good illumination conditions. We do not consider low illumination because almost no image data can be produced by the digital camera to be efficiently utilized by the system,

Detection ratios are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Normal Orientation(0 degree)** | **Orientation(90 degree)** | **Orientation (180degree)** | **Orientation(270 degree)** | **Average Detection** |
| **Plain Background** | 80% | 70% | 80% | 70% | 75% |
| **Complex Background** | 60% | 40% | 60% | 40% | 50% |

**Table 7.2: Orientation Analysis**

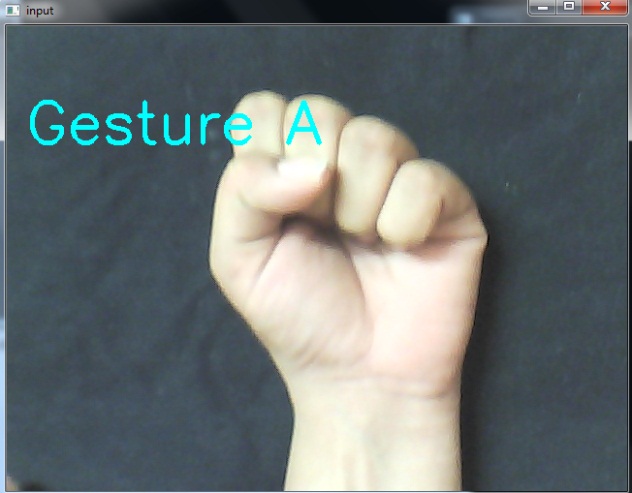
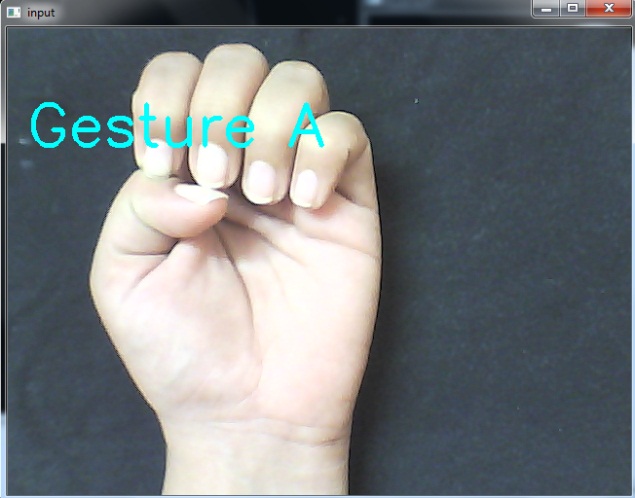
**Distinguish between similar types of gestures will give false output – possibility 50%**

For example: Observe alphabets A, E and S.

Alphabet A Alphabet E Alphabet S

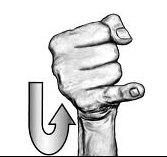
****

****

**Figure 7.10:Similar gestures.**

**Cannot recognize the dynamic gestures**

**For example:** Observe alphabets J and Z

** **

Alphabet J Alphabet Z

**Figure 7.11:Dynamic gestures.**

**Chapter 8**

**TESTING**

**8.1 Test Cases**

**8.1.1 White Box Testing**

In white box testing make sure that the system that built is being fully exercised, thus ensuring that it is performing the functions for which it has been programmed. This system has interfaced various methods and make sure that this interface is visible to the user. Here main and sole aim is to exercise all the technical capabilities of the system that built.

**8.1.2Black Box Testing**

In black box testing test the system in a random manner. Done this for some random functionality and depending on the output that is obtained a conclusion can be made as to whether the system that has been built is accurate or inaccurate.

**8.2 Test Case Implementation**

**8.2.1 White Box Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unit/Function** | **Loop/Iteration** | **Functionality** | **Subroutines** | **Efficiency** |
| rotateImage() | Iteration | To rotate an input image and provide the rotated image as output. | If loop is used and rotation is performed based on the rotation matrix concept. | Good efficiency with low time consumption because of simple structure utilized. |
| computeDistTransform() | Iteration with individual loop. | Computes the distance transform of the image providing concentrated pixels near the core axes of the image. | If loop is used to clone image for backup copies if needed. | High efficiency but memory utilization increases as the iteration progresses in time. |
| main() | Iteration with many loops. | Executes the entire program taking into consideration all the loops. | An infinite for loop, finite for loop and two if loops are used for video capturing and image processing. | High but tends to increase load on the system as the execution progresses. |
| for(;;) | Single infinite loop. | Enables the capture of multiple frames. | If loops for image processing, for loop for threshold of image pixels. and while loop for image inputs. | Good efficiency for the time it runs. |
| while() | Single finite loop. | Enables multiple input image acquisition. | None. | Excellent efficiency without stressing the system due to fixed memory usage. |
| for() | Single finite loop. | Enables acquisition of 2D matrices from template and input. | for() for y-axis component of images and if loop for incrementing count. | Moderate Efficiency. |
| Nested if else loop. | Single finite loop. | Detects the preprocessed image and compares with templates based on difference ratios and produces output. | If loops based on the number of inputs. | Low efficiency due to the ambiguity in calculation of count parameter. |
| else() | Single finite loop. | It sets the count parameter back to zero or increments it based on detection success. | Nested if loops for various count parameter conditions. | Ambiguous efficiency based on count parameter. |

**Table 8.1: White box testing.**

**8.2.2 Black Box Testing**

Black box testing as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case** | **Case**  **Description** | **Action** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| Initialization of the program through GUI. | GUI is used to initiate the correct module of the system. | The User clicks on the button to the program module he/she wishes to run | The correct module is run based on the button clicked. | The requisite module runs. | Pass |
| Gesture Input | A gesture needs to be provided by the user so that the features can be extracted for further processing. | The user provides a gesture with his hand to the camera.(according to a particular template). | Display a hyphen on the top left corner of the screen (proof of detection as per to code logic). | A hyphen is displayed on the top left corner of the screen. | Pass |
| Processing | The detected input needs to be further processed in stages:  1. Image Linarization.  2.Edge Detection  3. Orientation &Feature extraction | The gesture detected will be converted to binary, edges will be detected, template will be re oriented to match input and the hand contours will be extracted. | The contours are extracted accurately. | The gesture contours are detected. | pass |
| Gesture Output | The gesture processed so far needs to show the accurate name on the output screen. (Also considering orientation). | The processed image is passed on to the output window automatically once processing is done. | The hyphen on the top left corner is replaced by the detected gestures name. /symbol. | The name/symbol is displayed replacing the hyphen in the top left corner of the screen. | Pass |

**Table 8.2: Black Box testing.**

**Chapter 9**

**CONCLUSION AND FUTURE WORK**

1. **CONCLUSION AND FUTURE WORK**

**9.1 Conclusion**

The distinguishing factors of our system are that no additional hardware would be required externally by the user, the hands need not be perfectly aligned to the camera since they will be re-oriented to zero degree angle.

Many existing gesture recognition systems are available which provide excellent user-system interactivity by accepting user gestures through multiple ways. The cost though is high and the systems don’t usually cross all technical hurdles like use of single camera or any external hardware usage.

Sign language recognition system is an attempt to bridge the gap between the able and disabled human beings by providing simple yet efficient gesture recognition system.

SLRS is also not a perfect system in itself though, the effect of background uniformity and illumination do affect the system accuracy up to some extent.  But it successfully covers all the major concerns in the existing systems.

The factors like elimination of additional external hardware like sensor gloves or light emitting gloves, elimination of multiple camera usage by re-orientation of template images to zero degree angle thereby removing the need of providing user input in a properly aligned manner has been done successfully.

**9.2 Future Work**

1. Integration in cell phones having a front camera, wherein the  
   gestures can be converted into text/voice format thus enabling an end  
   to end communication between a deaf/dumb person and the normal person.
2. Multiple hand gesture recognition can be embedded in which both  
   the hands can be used for conveying a gesture and thereby implementing  
   dynamic gesture recognition.
3. Slides while presentations can be controlled by providing gesture  
   inputs for transition from one slide to another.
4. Hand gestures once detected can further be embedded into games  
   thus converting them into motion based games.
5. Computing systems can be managed, controlled and secured by  
   providing a set of gestures. Basic operations like starting, stopping  
   and switching between standalone applications thus improving the  
   existing Human machine interactivity.

**References:-**

**[1]Hand gesture recognition algorithm based on grayscale histogram of the image.(2010)**

**Authors:**RustamRakhimovIgorevich, Pusik Park, Dugki Min, Yunjung Park, Jongchan Choi, Eunmi Choi.

**[2]Real-time Sign Language Recognition based on Neural Network Architecture.(2011)**

**Authors:**PriyankaMekala, Ying Gao, Jeffrey Fan1 ,AsadDavari

**[3]Hand Gesture Recognition in Images and Video**

**Authors:**Ilan Steinberg, Tomer M. London, Dotan Di Castro

# [4]A background subtraction method using color information in the frame averaging process

**Authors:** Chaudhari, A.Sch of Electr Eng., Univ of Ulsan, South Korea Sang-jin-cho; Ui-Pil-Chong

**[5]An adaptive real-time skin detector based on Hue thresholding: A comparison on two motion tracking methods.**

**[6]** [**http://docs.opencv.org**](http://docs.opencv.org)

**[7]Learning OpenCV: Computer Vision with the OpenCV Library.**

Author: [Gary Bradski, Adrian Kaehler](http://shop.oreilly.com/product/9780596516130.do#tab_04)

Publisher: O'Reilly Media

**[8]OpenCV 2 Computer Vision Application Programming Cookbook**

**Author(s):**[Robert Laganière](http://www.packtpub.com/authors/profiles/robert-laganiere)

Publisher: PACKET.

**Appendix A**

**Paper presented:-**

Presented a technical paper on **“Sign Language Recognition System”** at **“National conference on Computing and Information technology (NCCIT).”**



****