GESTURE RECOGNITION-BASED CONTROL OF MOBILE ROBOT

END- SEMESTER REPORT

Submitted by:

ANIKET SAMANT 2012A8PS354P

TANMAY AGARWAL 2012A8PS500P

Under the supervision of:

Dr.Surekha Bhanot

In partial fulfilment of the requirements of the course:

INSTR F367 – Laboratory-Oriented Project

December 2015

Acknowledgements

We would like to express our deepest gratitude to Dr. Surekha Bhanot ma'am for fostering our interests in the field of Human Computer Interaction, a sub-field of Computer Science. We are greatly indebted to her that we got this opportunity to explore this field. Your genial teaching and guiding style piques our curiosity and inspires us to learn.

We would also like to thank the staff of the Electronics and Instrumentation Department for providing the necessary help while working on the robot in the lab.

Table of Contents

Introduction

Gesture recognition is defined as the mathematical interpretation of a human motion by a computing device which could take inputs from a still-image camera (frames) or a video recording device. Using human gestures as a means of interaction with computing devices has picked up pace in recent years with the advent of platforms such as the Xbox Kinect which has made development in this field considerably accessible to students and researchers alike.

Gesture recognition technology makes use of a camera that reads the movements of the human body and communicates the data to a computer that uses the gestures as input to control devices. For example, clapping of hands together in front of a camera can produce a trigger that can be used in software development when the gesture is fed through a computer.

One way gesture recognition is being used is to help the physically impaired to interact with computers, such as interpreting sign language. The technology can eliminate input devices such as mice and keyboards by changing the way users interact with computers and allowing the body (if disabled) to give signals to the computer through gestures such as finger patterns.

Haptic interfaces require the user to wear special equipment or attach devices to the body for functioning. Gesture recognition, on the other hand, involves use of gestures of the body to be read by a camera instead of sensors attached to a device such as a data glove. Such technology also can be used to read facial and speech expressions and eye movements.

Gesture recognition, along with facial, voice, eye tracking and lip movement recognition are components of perceptual user interface (PUI). The goal of PUI is to enhance the efficiency and ease of use for the underlying logical design of a stored program.

Applications of Gesture Recognition

The applications of gesture recognition are immense and find relevance in many upcoming fields (such as AI) and can also be applied to many fields that have been in use for a long time. Some of the applications are given below.

Home automation

Gesture recognition systems can be used to identify gestures such as waving arms, punching, and kicking, and control home appliances (connected by a common protocol such as WiFi) to provide a completely automated home experience. For example, switching off of the lights automatically when no gesture is detected for a long period of time can help reduce consumption of electricity. Similarly, controlling fans, lights, televisions and other such appliances using gestures can increase the degree of convenience.

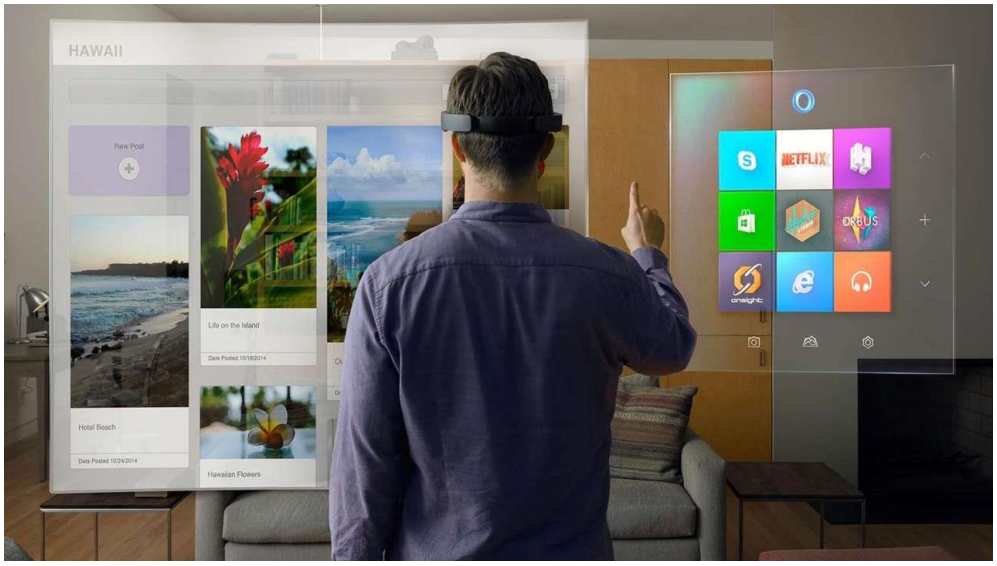


Fig1: The 'Ninja Sphere', a gesture recognition based home entertainment system

Driving assistance

Recently manufactured vehicles have gesture recognition based driving assistance systems which can be used to control the infotainment system, air conditioning, and lighting in the automobile. But the most important application lies in safety of the passenger(s). Eye-tracking systems can tell the driver if they are drowsy and not in a

state to drive and can bring the vehicle to a stop in a safe manner. This could potentially save thousands of lives as a majority of road accidents occur as a result of drivers not being attentive while driving.

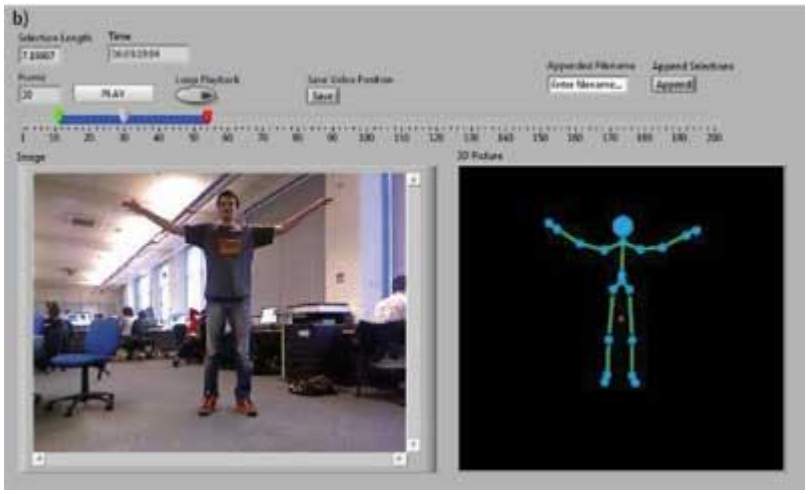


Fig2: The Xbox Kinect's recognition system's output

Control of robots

Gesture recognition can also be extended to the field of robotic control. The Microsoft Xbox Kinect platform is particularly of great use in this field – robots can have mounted units of the Kinect which can recognize user-defined gestures and perform functions as desired by the user, which could involve transport, services such as cleaning, sorting, etc.

Sign Language Recognition

Since hand gestures are used in interpretation and explanation of the subject in the sign language, it has caught major attention from the researchers. Different gestures have been proposed to convey different meaning in the sign language. MLP neural network, boundary histogram and dynamic matching are some of the algorithms which have been implemented for hand gesture based sign language recognition.

Gesture Recognition for Robotic Control

We have assembled a mobile robot (four-wheeled) whose direction is controlled by an Arduino board and the corresponding motor drivers. The input to control the robot is provided through user gestures, which are interpreted by the computer using gesture recognition algorithms implemented using the OpenCV image processing libraries and the control signals are sent to the Arduino board accordingly via serial communication.

Software

The OpenCV libraries to be used for gesture recognition have been used using the

- Microsoft Visual Studio platform: This platform is very versatile and is used for various kinds of software development in C++, C# and Visual Basic. For this project, we found the OpenCV libraries relevant which can be easily integrated and used with MS Visual Studio. This library provides various algorithms for image processing, object recognition and computer vision. We use this library to develop our software application for detection and

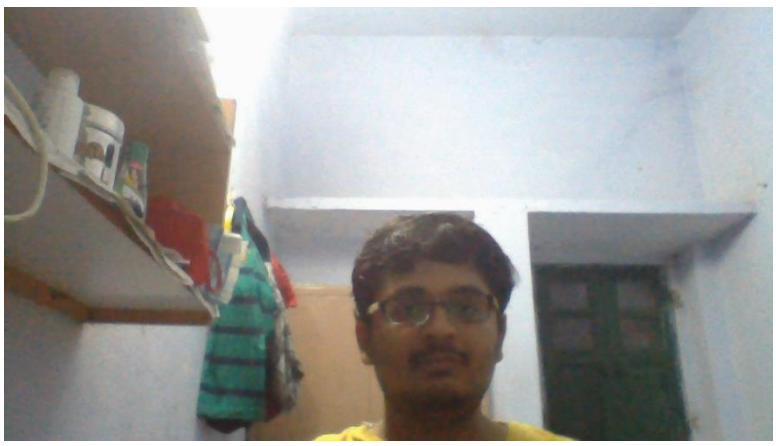


Fig3: Original image, acquired using webcam

recognition of hand gestures.

- Visual Micro plugin – This is a plugin for controlling Arduino on Microsoft Visual Studio.

Fig4: Canny edge detector output produced using OpenCV library

- Arduino IDE – It is a standard editor for microcontroller based applications. We use this IDE for preliminary testing of Arduino (AtMega 2560) board and for analysing the motor response to different input signals.

Hardware

- We have used an Arduino board to control four DC motors (12 V, 60 RPM, 800 mA NL / 4 A FL current).
- We have used the concept of a differential drive to control the direction of the robot. (Directions being forward, reverse, rotate left, rotate right and controlled using inputs to L293D motor driver ICs)

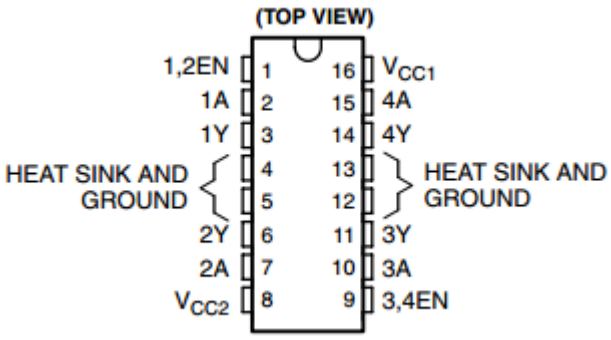


Fig5: Pin-out of an L293D IC

- Two TI L293D motor driver ICs have been used, and one set of input control signals is common to each (since the two wheels on one side will have the same direction at any given point of time).
- We have used the laptop's in-built webcam for testing the OpenCV libraries though using an external camera mounted on the robot will yield similar results.
- The gesture recognition algorithms are performed using the laptop (MS Visual Studio) and the corresponding control signals are sent to the mobile robot using the serial communications port through the USB serial connector.

Gesture Recognition Algorithms

There are multiple ways to approach gesture recognition – it could be algorithmic or even training model-based. Here are some of the algorithms that are relevant to our project:

- **Identification of number of fingers being shown to the camera by the user:** This algorithm aims to find the connected components including the hand and fingers and the centre of this single component (The image is first converted to grayscale). A circle of a radius less than the distance between the centre and the tip of the middle finger (i.e. usually 0.7 times the distance) is drawn around this centre and then the number of transitions between black to white and white to black are counted which gives us the number of fingers being shown by the user.

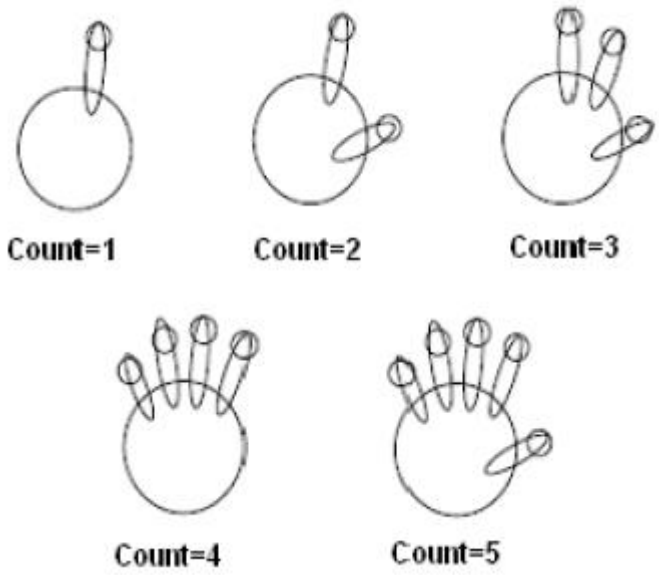


Fig6: Illustration showing the algorithm to count number of fingers

- **Neural-networks-based gesture recognition techniques:** There are several algorithms based on neural networks that can be used for gesture recognition; most of them rely on the hue of the hand being different from that of the surroundings so as to set a threshold and separate the image of the hand from the surroundings'. The gesture is interpreted using this image and

the neural network is trained using such images so as to classify the required gestures with greater accuracy each time in order to make the classifier reliable. The neural networks have an added advantage unlike other classifiers that it can incorporate noisy backgrounds which sometimes may incorrectly be a contestant of hand gesture.

- **Hand Recognition based on Haar Cascades (Machine Learning based approach):** This method was proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning approach where a cascade function is trained based on positive and negative sample images. This cascade function is technically a feature classification support vector machine (SVM) classifier or a boosted classifier. Based on these images, the model's false detection rate decreases as more and more number of samples are given for training. The training data consists of positive samples of the object, in our case, our hand, given in different gradient, orientations and scale. The negative samples images include everything in the environment except the object to be recognized.

Algorithms Implemented for Hand Recognition

To ensure that we get the best possible results while detecting hand gestures to control the robot, we need to detect the hand correctly before moving on to detect the gestures indicated by it. Thus, we implemented several algorithms and eventually created one that works well even in a dynamic background to detect the hand effectively while separating it from the background.

Pixel Intensity Values-Based Gesture Recognition

This algorithm makes use of the conversion of the incoming video stream from the default RGB values to grayscale. It relies on the fact that the intensity of the pixels covered by the hand gestures is different from the background's. This method depends highly on the lighting conditions around the user and on the nature of the background (that is, it must be absolutely static and very distinct in intensity as compared to the hand). It cannot be used when the background is dynamic or when the intensity values of the background approach that of the hand's.

The algorithm followed here is:

1. Conversion of input video stream to grayscale
2. Comparison of pixel values with a certain threshold value (depending on the background, it can be set to a particular value)
3. Creation of a binary image based on the user-set threshold value
4. Application of gesture recognition algorithm on the image so generated
5. Control of the mobile robot through serial communication

Since this algorithm fails in the case of a changing background, or even one that is not uniform, we cannot use it for an application demanded by a mobile robot. Hence, this algorithm has been discarded from use in our project.

Background Subtraction-Based Gesture Recognition

This method is a step ahead of the aforementioned method in that it takes into consideration multiple backgrounds (which may not be plain in texture). The background has to be static though; a constantly changing background (changing after intervals of even a few seconds) can yield errors in gesture recognition. The algorithm followed in this case is:

1. Storing the background image (without the presence of the hand, or any other object that will not be in the same position during the course of the process) after choosing a suitable background for performing gesture recognition.

Fig7: Background image being used for the background subtraction algorithm

2. Conversion of this captured image to grayscale.
3. Starting the input video stream and converting it from RGB to grayscale.
4. Subtraction of the pixel intensities of the initially captured grayscale background image from the continuous grayscale input frame.
5. Considering the absolute values of the resulting matrix (image) and converting it to a binary one using a suitable threshold value.
6. Application of gesture recognition algorithms on the resulting binary image.
7. Serial communication to the robot.

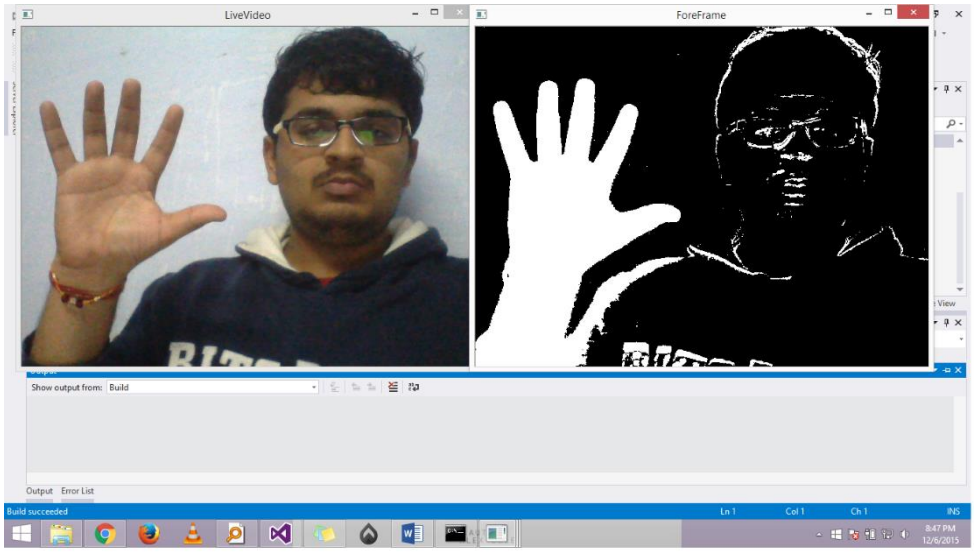


Fig8: Binarized image output from Background Subtraction Algorithm

This method works quite well in case of static cameras, as any difference in the captured video stream from the background is detected very easily and accurately. Lighting conditions are not a major concern so long as the background remains stable and the hand does not cast a considerably dark shadow on it so as to affect the pixel intensities of the background to a large extent. Stray objects entering the video frame could also lead to errors. But since this project involves a mobile robot, the background is expected to change very frequently. Hence, this method has been discarded from our options, though it is quite an effective one in case of static backgrounds.

Haar Cascade-based recognition

This method involves the use of machine learning algorithms on the input video feed to identify certain patterns that are trained to be recognized with the aid of Haar Cascades. In short, it works on the concept of Support Vector Machines (SVMs) in which the cascade network is trained using positive and negative samples in order to recognize similar samples in the input video in real time.

The algorithm followed here is:

1. Training of Haar Cascade using positive and negative samples and generation of corresponding .xml file.
2. Using the network on the input video stream to detect trained patterns in it. These patterns could include the palm, the fist, etc.
3. Application of gesture recognition algorithms once the patterns corresponding to these gestures have been identified.
4. Control of robot through serial communication.

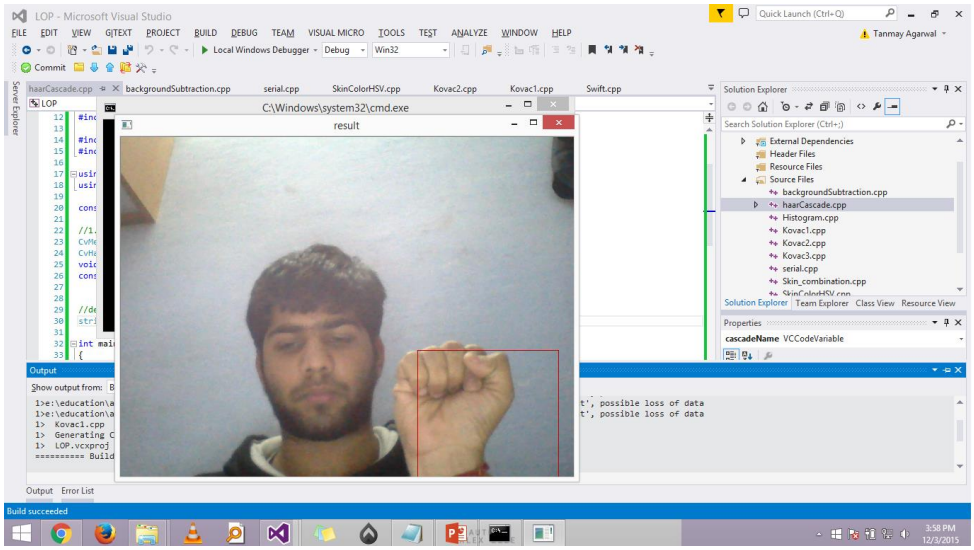


Fig9: Fist detection based on Haar Cascade Learning approach.

This method is a good approach as it does not depend heavily on lighting conditions or on the background. It does not even require that the input video stream be converted to the grayscale space, as is the case with the other methods discussed. Since it is a learning-based approach, it can be trained for several orientations and can be modified easily. But it is highly error prone if the number of positive samples provided is less or if the required correct orientations are not provided in the positive sample space. During testing, even the slightest resemblances to the desired gestures (false positives) were detected as valid gestures. Hence, due to reliability issues, this method was tested but not implemented on the robot and further algorithmic approaches were explored.

Codebook algorithm

This algorithm was tried along with the Haar Cascade one as part of the learning-based rather than algorithmic approaches. It involves training of a network to recognize the background beforehand and effectively subtract it from the desired foreground gestures. The basic algorithm followed is:

1. Training of network to identify background (static); time required depends on the number of samples.
2. Capture of input video stream and usage of trained network to subtract background.
3. Conversion of resulting foreground image to grayscale.
4. Conversion of the resulting grayscale image to binary and application of gesture recognition algorithms on it.
5. Control of robot using these algorithms.

Fig10: Codebook algorithm output showing the convex hull formed around the hand and the binary image.

This method has a main drawback that the background has to be almost static. Even if the background colour changes slightly due to a difference in lighting conditions, or shadows, or an undesired object(s) enters the background frame, the network has to be trained again to eliminate such disturbances. Repeatedly training the network

leads to a constant break in the robot operation, and hence is not preferred, and algorithmic methods are followed.

Skin colour-based detection

In the previous methods, hand gestures are detected either through trained networks or by using background subtraction methods. There are certain advantages of using those methods over the forthcoming method but the disadvantages severely reduce their reliability as compared with the latter. This method can be used in dynamic, considerably noisy environments too, as opposed to the previous ones.

As the name suggests, skin colour-based detection involves using the RGB/HSV values of the pixels to determine if the pixel has a skin-colour value. This information can be used to recognize gestures and control the robot accordingly. The algorithm used is as follows:

1. Capture of input video stream and accessing of pixel information (RGB or HSV).
2. Classification of input video pixels depending on RGB or HSV values in the following manner:
    a. RGB: Using the Kovac, Swift or Histogram model in order to determine if the pixel is skin coloured or not. In this project, after due testing, we have concluded that the Kovac model works best for our application and have thus used it to detect skin-colour pixels.

Rule 1: $R > 95$ and $G > 40$ and $B > 20$ and
Rule 2: Max (R, G, B) – Min$(R, G, B) > 15$ and
Rule 3: $|R - G| > 15$ and
Rule 4: $R > G$ and $R > B$

Fig11: Rules of the Kovac based on which a skin colour pixel is detected.

b. HSV: A certain range of H, S, and V values can be considered as being skin colour, but it is a very crude approach and is quite error prone as any pixel which lies in the HSV range may be classified as a skin colour pixel. It is quite accurate, though, and may be considered as an alternative approach to skin colour detection apart from the Kovac model.

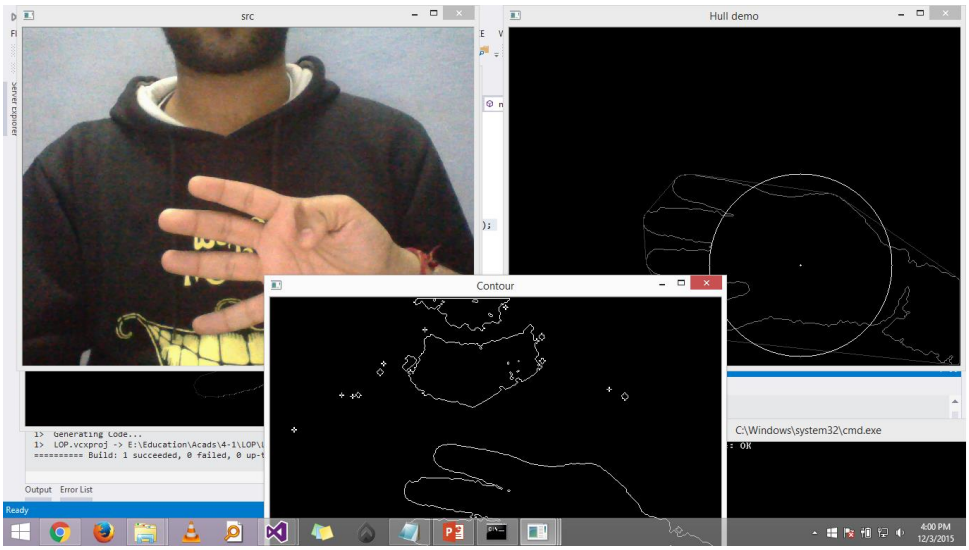


Fig12: The top left image shows the live feed from the camera. The middle frame shows the HSV based skin colour segmentation. The right frame shows the contour having the maximum area with its centre and a circle around it.

3. Using the skin-colour threshold to binarize the image into skin-coloured and non-skin-coloured regions.
4. Finding the contours which are contained in this binary image.
5. Elimination of contours of relatively small areas in order to leave only the largest area contour into consideration. This is done using OpenCV functions.
6. Draw the largest area contour. Then find and draw the convex hull of this contour to get a convex polygon which contains the contour inside it.
7. Determination of the centre of this contour using inbuilt OpenCV functions.
8. Application of gesture recognition algorithms based on the relative position of the fingers to control direction of robot.

This method is quite robust in the sense that it can be used in a dynamic background as long as there do not exist large patches of skin-coloured objects in it. It can also differentiate between hand-shaped objects and the actual hand (considering the objects to be other than skin colour) and thus avoids errors due to the presence of undesired objects in the background. If light of sufficient intensity is provided in front of the camera so as to illuminate the hand adequately, then the background lighting conditions do not affect gesture recognition much and hence this method can be used in poor lighting conditions too. The only drawback is that if a skin-coloured object of considerable size, or a skin-coloured wall is present in the background, the algorithm wrongly treats them as part of the hand and yields wrong results. But such conditions are improbably and hence, for all practical purposes, this method serves the required application best, and hence is the method we have adopted as the final one in order to control the mobile robot. The algorithms used to control the robot will be dealt with in the next section.

Algorithms Implemented for Gesture Recognition

We have implemented nine kinds of gesture which can be used to control our mobile robot.

The first set of gestures includes the finger count based gestures which controls the bot based on the finger count. Finger count of 1 makes the bot move forward while that of 2 makes the bot move backward. Finger count of 3 turns the bot leftwards and 4 turns the bot rightwards. Finger count of 5 can be used to stop the bot. The default count of 0 keeps the bot stationary. Any count of greater than 5 is an error and hence it is ignored and not sent to the bot through serial communication.

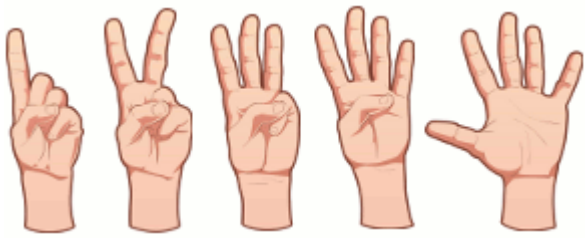


Fig13: Finger count based gestures.

To detect the finger count, we implemented two approaches.

1. Transition based approach – In this approach, we iterate over the entire contour points to count the number of transitions that occur when we move in and out of the circle (with radius 0.7 times the distance between the centre of the contour and the middle finger). The count of the fingers can then be easily computed based on these transition count.

2. Defects Points approach – In this approach we try to find the defect points (points which lie inside the hull on the contour) of the contour. For this convexityDefects function is being used which returns all the defects points pixels. Based on these defects points, we can easily find the number of fingers.

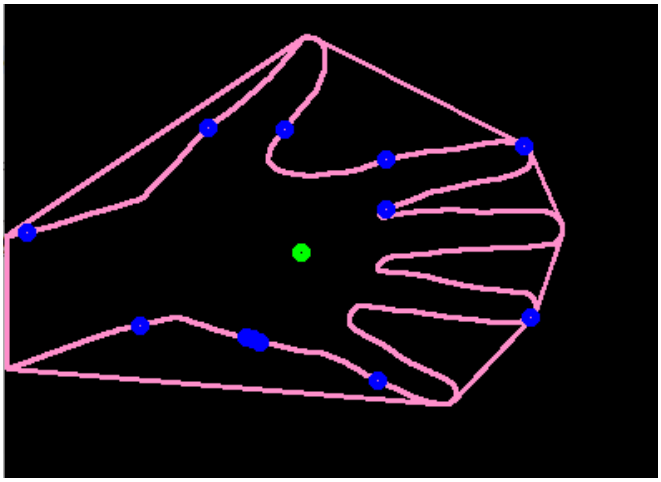


Fig14: Convexity Defects output when applied on hand contour.

Based on the finger count, we recognize the gesture. Each gesture is thus represented with a character (with integer value of this character being the finger count). Thus, a character-to-gesture look up table is generated. This character is then sent to the Arduino UNO via the serial communication port at a baud rate of 9600bps.

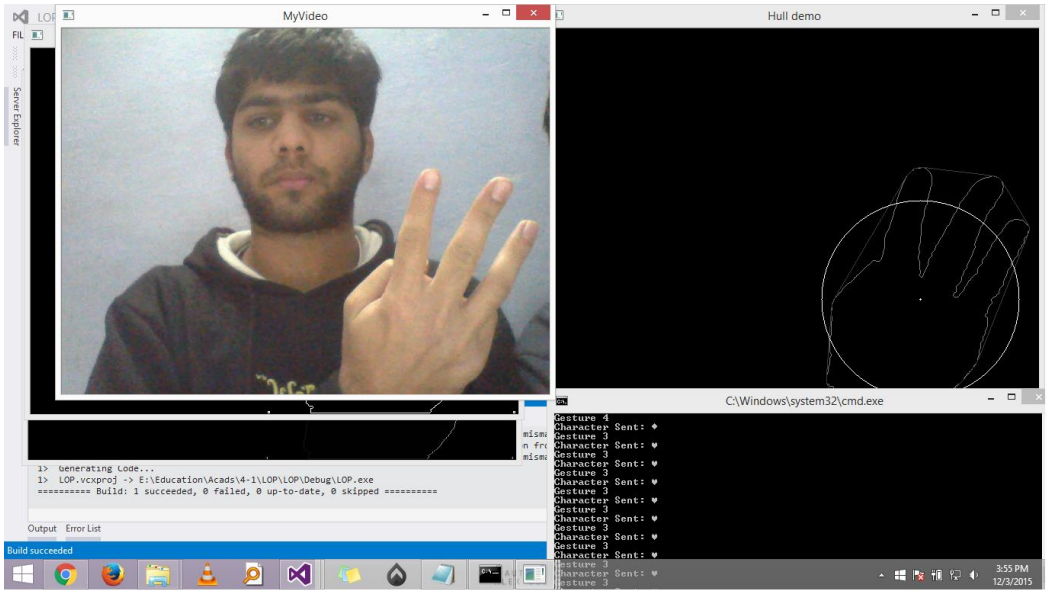


Fig15: Finger count gesture: The left image shows Gesture 3 in the live video frame whereas the right one shows the hand contour with the circle and the centre. The lower window displays that gesture 3 has been recognized and correspondingly a character is sent from the computer to the Arduino board.

The second set of gestures include the direction based gestures. These gestures are based on the relative position of the finger with the centre of the hand. An up gesture makes the bot move straight whereas the down gesture moves the bot back. Similarly, left gesture turns the bot left and vice-versa.

Fig16: Direction based gesture: Up and Down Gesture

To recognize these gestures, we try to check if there is any contour pixel that lies at a certain distance from the centre and in a direction relative to the centre of the hand contour. We implemented this logic and took it one step ahead to look for the farthest pixel from the centre and determine its relative position from the centre.

A similar kind of look up table is generated as in finger count approach based on the gesture. Thus every gesture is modelled as a byte character which is then sent to the Arduino UNO board via serial communication port at a baud rate of 9600bps.

The image shown below shows the output of the Direction based gesture recognition.

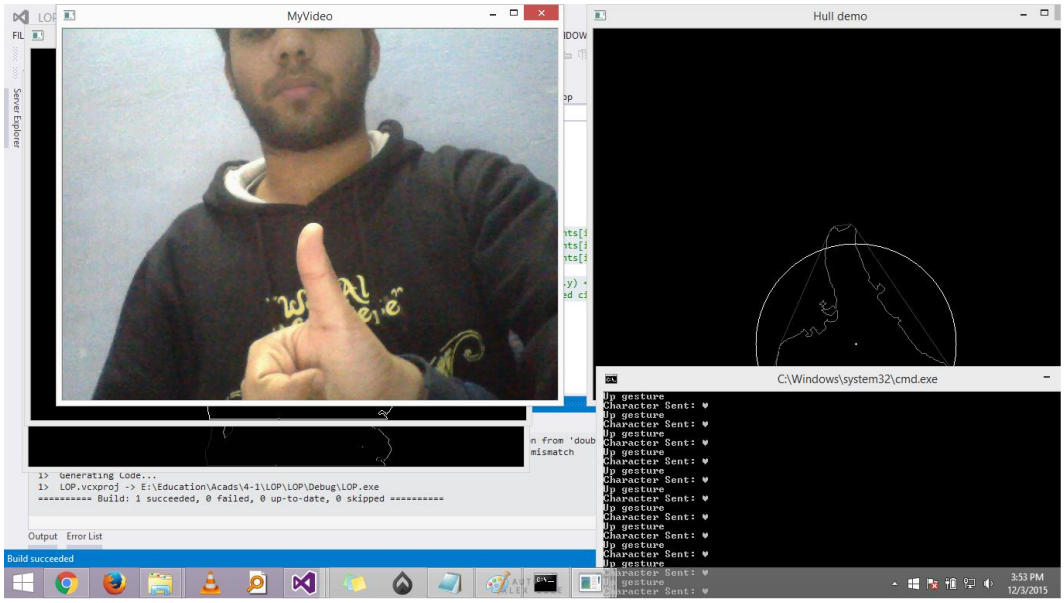
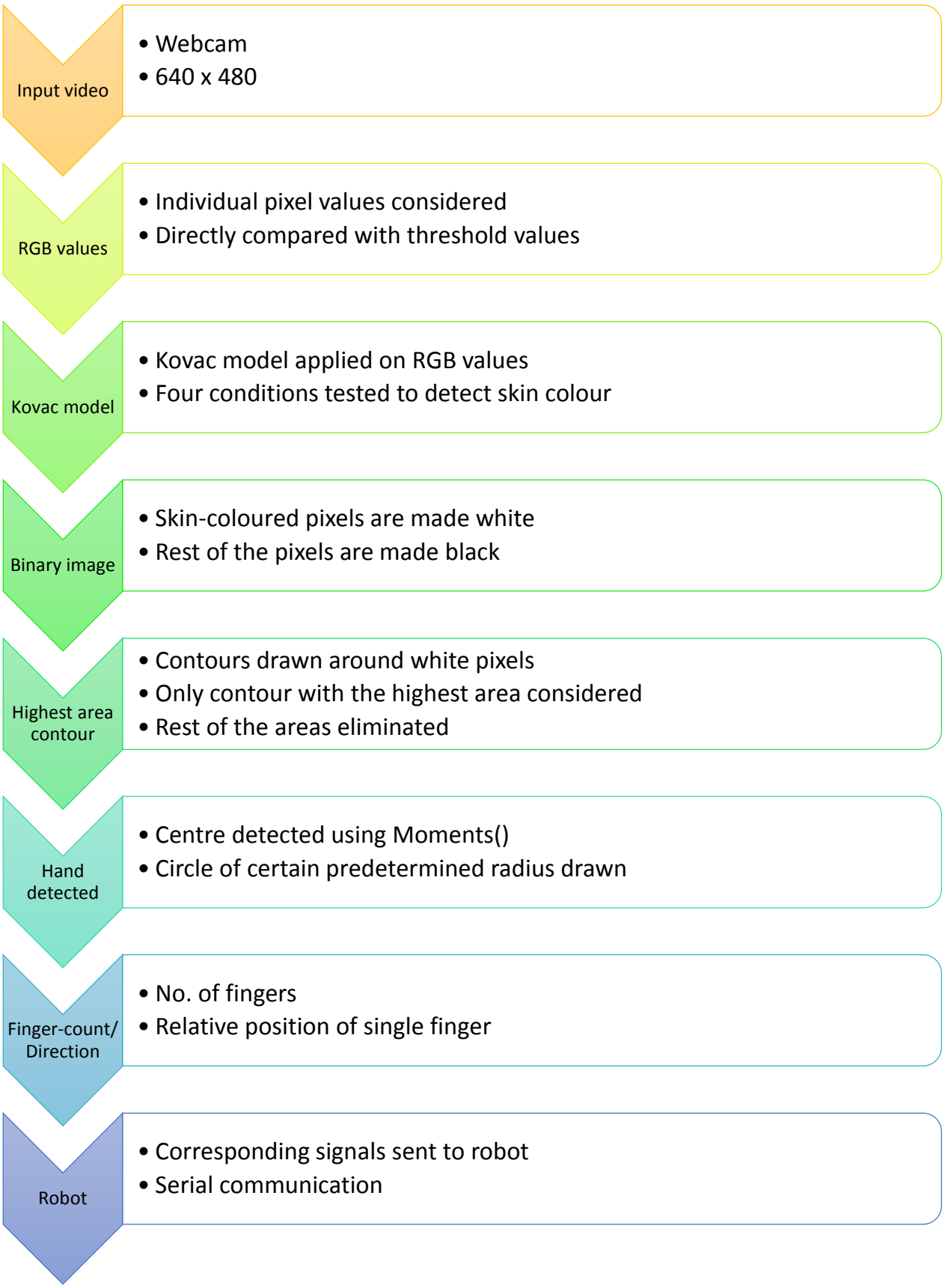


Fig17: Direction based gesture: The left image shows Up Gesture in the live video frame whereas the right one shows the hand contour with the circle and the centre. The lower window displays that Up Gesture has been recognized and correspondingly a character is sent from the computer to the Arduino board.

Final Algorithm Flowchart



Conclusion

The algorithm that we have proposed is fast and simple for the hand gesture recognition problem. The problem takes in real time images and segments the hand region. It is also seen that these segmentation algorithms are computationally efficient when compared to other algorithms which involve training of a classifier or other complex methods. The algorithm can handle dynamic conditions and works well even the camera is moving.

For the problem of controlling a mobile robot, we have considered only a limited set of hand gestures. Our algorithm may be expanded to recognize other sets of hand gesture as well. The segmentation problem that we have used is very simple and would need to be improved if the condition are more challenging.

Apart from the segmentation problem, the learning based approach is a more trustable algorithm as it will recognize only those object (i.e. hand in our case) which have been used during the learning stage. But the algorithm is computationally very expensive as it involves training over a large dataset of positive and negative samples.

Finally we have also noticed that neural networks are seen to give the best output for the hand gesture recognition problem. It is an adaptable algorithm which adjusts itself in the presence of noisy environments. It gives a good accuracy with very few false positives and false negatives but the training is computationally expensive in terms of resources.

References

1. Robust Hand Gesture Recognition Algorithm for Simple Mouse Control, Vivek Veeriah J. and Swaminathan P. L., *International Journal of Computer and Communication Engineering, Vol. 2, No. 2, March 2013*.
2. A fast algorithm for vision-based hand gesture recognition for robot control, Asanterabi Malima, Erol Özgür, and Müjdat Çetin.
3. Hand Gesture Recognition using Neural Networks, G.R.S. Murthy and R.S. Jadon.
4. A Static Hand Gesture Recognition Algorithm Using K-Mean Based Radial Basis Function Neural Network, Dipak Kumar Ghosh and Samit Ari.
5. Online Hand Gesture Recognition Using Neural Network Based Segmentation, Chun Zhu and Weihua Sheng, *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems October 11-15, 2009 St. Louis, USA*
6. Static hand gesture recognition using neural networks, Haitham Hasan and S. Abdul-Kareem
7. A Neural Network based Real Time Hand Gesture Recognition System, Tasnuva Ahmed
8. Face Detection using Haar Cascades, (http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0)
9. Cascade Classifier Training (http://docs.opencv.org/doc/user_guide/ug_traincascade.html)
10. Haar Feature-based Cascade Classifier for Object Detection, (http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html)