# Robot Control using Pattern Recognition

*Project report*
*submitted*
*in partial fulfilment of the requirement for the*
*degree of*

**Bachelor of Technology**

By

**Aritra Bhattacharyay (1505016)**

**Debaruna Saha (1505029)**

**Rohan Saha (1505614)**

Under the supervision of

**(Dr.) Prof. Sudan Jha**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**
**Deemed to be University, BHUBANESWAR**
**APRIL 2018**

# CERTIFICATE

This is to certify that the project entitled **"ROBOT CONTROL USING PATTERN RECOGNITION"** submitted by

| | |
|---|---|
| **ARITRA BHATTACHARYAY** | **1505016** |
| **DEBARUNA SAHA** | **1505029** |
| **ROHAN SAHA** | **1505614** |

in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the work carried under my(our) guidance and supervision at School of Computer Engineering, KIIT Deemed to be University.

Signature of Supervisor

(Dr.) Prof Sudan Jha

School of Computer Engineering

KIIT Deemed to be University

**The Project was evaluated by us on _____**

EXAMINER 1

EXAMINER 2

EXAMINER 3

# Acknowledgement

We feel extremely privileged in expressing our sincere gratitude to our supervisor and mentor (Dr.) Prof. Sudan Jha, for his constant guidance and supervision throughout our project work. His way of solving the real world problem by implementing it in machines have inspired us to take up this project. Our heartfelt thanks to you sir for the support and patience shown to us.

We are also thankful to our department of School of Computer Science for giving us this opportunity to implement our idea into real time implementation.

Aritra Bhattacharyay (1505016)

Debaruna Saha (1505029)

Rohan Saha (1505614)

# ABSTRACT

Robotics has paved the way for automation in diverse areas in the industry. Over the years, people have come up with methods to implement automation into robotics and thus implement industrial based automation. Methods to incorporate human computer interfacing have led to ease of use of systems besides hard coding algorithms into systems to execute certain tasks.
Interfacing systems with humans helps to control devices and automatic systems with natural human intuition and thus elevate user friendliness. Some common examples of intuitive interaction are hand gestures, audio commands and touch-screen interaction to name a few.

Design and implementation of a pattern recognition algorithm is carried out which detects the number of fingers in the specified region of interest in the captured frame. We have not considered the application of neural networks in the project due to its massive training time and also due to the scarcity of training data available. We have used a mathematical approach to solve the problem of finger detection. Rule of cosines for a triangle is used to detect the number of fingers in the image frame by determining the angle between the ridge between each pair of fingers. A standard angle is used to count the ridges and subsequently, the number of fingers. The standard angle is determined with the help of trial and error. The number of fingers detected is then mapped to an event which then makes a robot manipulator draw shapes having sides equal to number of fingers. We have used the Python' programming language to simulate the system because of the rich set of features it provides for solving problems pertaining to computer vision. Two separate files have been created, one containing the functions to carry out respective events (drawing shapes) and the other file containing the main script which includes contents from the first file. This helps to preserve modularity of the system. RoboDK software has been used to simulate the system so as to alleviate the costs which would have been incurred in case a physical model was a required to develop. RoboDK is a software which provides an Application Programming Interface (API) to control the movement of the end effector of the robot manipulator using the python programming language. To overcome the problem of accumulation of commands a timer is made to run between the start of the event and the end of event. In addition, the statistical concept of mode is used to determine which pattern is recognized the maximum number of times in the frames captured in the time interval where the robot is performing some operations, in this case, drawing shapes. The results obtained may not be hundred percent accurate but are reasonably accurate and does the work of detecting fingers smoothly. Though implementing convolution neural networks could have achieved state of the art results, solving the problem using a mathematical model helps to process the frames faster and also helps cut down on the training time required for a neural network; the model developed in this paper may not provide accurate results in complex applications but can be used in simple applications like controlling a robot manipulator.

# TABLE OF CONTENTS

# LIST OF FIGURES

# Chapter 1

# Introduction

Robotics is a field of study that is into existence from a very long period of time. It was present even during the start of the Egyptian civilization. In those days, robots had a very little functionality and capability and were mainly dependent on human intervention physically. In the present day robots and robotics has evolved a lot and it is one of the most lucrative fields in all the disciplines of engineering. Moreover, for proper functioning of a robot, knowledge from diverse branches is required.

Today the robots are being provide with intelligence and also better communication and efficient working. So, keeping this in mind we have come up with an idea to develop an algorithm for a robot so that the communication is through hand gestures.
A gesture is a movement of a part of the body to represent an idea, task or a meaning. Now we could have even used normal voice communication or remote devices to control a robot but why gestures. The remote devices are good and serve the purpose but there will be only one remote device to operate a robot and a remote device is not good for long term use as the electronic components become faulty after using it for long period of time. Using voice communication is not very useful in case of small robots that can be used in homes or for commercial purpose. As it is very difficult for a robot to understand English for achieving communication through voice we need to convert the audio signals into text and then do a lot of computation on the text for deciphering it and making it understandable to the robot. So, for these reasons we have chosen gesture recognition to be the best solution for this problem. In gesture recognition a robot a camera is attached to a robot or it is remotely located with some additional hardware for processing and identification of the gesture. The operating person displays different gestures with hand to signify the pre-defined task the robot will do for a particular gesture. The algorithm identifies the gesture and sends a signal to the robot to perform a particular task. In this way gesture recognition works in controlling or directing the tasks performed by a robot. In our system we are simulating the entire thing in a software, as our main aim is to develop an algorithm for appropriate pattern recognition keeping in mind that the scope of the application is for small robots with limited hardware support. So, in the construction of our system we have used a number of different approaches and finally come up with an approach that is accurate and also very less complex computationally. For demonstration purpose we have used only two hand gestures.

# Chapter 2

# Literature Review

In the field of gesture recognition there has been a lot of models proposed. The highest accuracy has been achieved by deep learning models that have been able to learn the temporal features of the images and the videos. According to the different models and techniques used the deep learning models can be divided into three categories: -

- 3D-Models
- Motion-based input features
- Temporal methods

In 3D models we are combining the spatial and the temporal features together. The spatial features are in 2D and there is another dimension of temporal features. The high number of parameters present in the 3D model makes the model very complicated and very difficult to train. So, many researchers have suggested different ways to train the 3D CNN model. One of the approaches refers to initialize the input weights from the 2D CNN model of ImageNet dataset. Another approach proposes that to divide a 3D CNN into sequential CNN where the 2D spatial features and the 1D temporal features will be learned in a sequence in the different layers of a CNN. To improve the accuracy of 3D CNN learning the convolutions can be performed on optical flow maps.

With respect to the problem of gesture recognition [1] has devised a very novel approach where the CNN architecture consists of seven layers. The first layer consists of a CNN layer, followed by a sampling layer, then again a CNN layer followed by a sampling layer. The second sampling layer is followed by a third CNN layer, two fully connected layers for output follows the third CNN layer. Now what operation is performed in the CNN layer after taking input from the CNN layer. The Microsoft Kinect provides the color video stream and also the depth video stream. The color stream contains three channels that are the Red, Green and Blue channel. The depth channel contains another two types of input that are the depth and the body skeleton. In total we receive five kinds of input data. For each visual source the authors consider nine frames of size 64*48centered on the current frame as input to the 3D CNN. The above computation gives us five feature maps each feature map contains nine frames stacked to form a cube.

In [2] the authors have also used a 3D CNN but the approach used by them is very innovative and new and part of the approach can be used for gesture recognition where there is limited hardware availability. The authors have proposed an approach for gesture recognition of 19 classes of gestures. The input to the neural network architecture is 57*125*32and the input has been divided into two parts the first part goes through 3D convolution and max pooling, the resulting dimensions are of 25*59*14. The first part is forms the input to a high resolution neural network and the second part which is of dimensions 25*52*32forms the input to a Low resolution neural network. The input to the low resolution neural network is achieved by down sampling the input by 2. The High resolution network and low resolution network each go through three layers of 3D CNN followed by max pooling. After which the outputs are sent to two fully connected layers and followed by a softmax layer. Two separate softmax outputs that are the probabilities are generated. These two softmax undergo element wise multiplication to form the output layer for the prediction of the 19 classes of gestures.

In [3] a Deep neural network has been proposed which is very effective in recognizing gestures from a sequence data like a video. Suppose there is a video and the video is divided into short clips and phases each phase is stacked up into frames and forms the input to a 3D CNN network the output of each of the 3D CNN is input to the RNN units of a RNN layer. Each RNN unit is followed by a softmax layer and finally the output of all the softmax units are accumulated in a connectionist temporal classification(CTC). The CTC is a cost function used in unsegmented streams of data.

The above methods are the most innovative methods that have been developed by the researchers and the computer vision community with respect to the context of Gesture Recognition. In the above approaches high performance neural networks have been used as the need of the hour in industry is very high accuracy. These networks are suitable for defense and military applications or places where very fine computation is performed. The above networks require very high processing power and are not suitable in places where there is limited hardware available and where the application is more offline and there is no connectivity with the internet. In these kind of situations, the above architectures cannot be used. For example, in homes and colleges or remote areas. So, for that reason we have come up with an approach where with the use of simple image processing we are being able to detect and identify hand gestures.

# Chapter 3

# Implementation of Pattern Recognition in Computer Vision

## 3.1 Overview

Here, we present our approach and implementation of the algorithm where we use the cosine rule for triangles to ultimately detect the number of fingers in the selected region of interest in the whole frame captured through a camera source.

In our approach, we first open up the inbuilt webcam to stream the picture frames to the program containing the main logic. Then we perform numerous operations on the captured frame detect the fingers in the end. The number of fingers detected in the selected region of interest in the captured frame is mapped to a specific action which the simulated robot carries out. The process of capturing frames and performing operations on each frame is set up in such a way that the whole process is inside an infinite loop. Currently, for demonstration purposes, we have implemented the algorithm to detect only four fingers and five fingers. The implemented code can be extended to incorporate detection of more number of fingers too. The process can be terminated by pressing a button on the keyboard signaling the end of the application.

## 3.2 Selection of simulation software

To alleviate the costs incurring in the development of a physical model consisting of hardware components, we decided that a software simulation can be of help to quickly test the algorithm and later deploy the model on a real physical system.

We used the RoboDK robot simulation software to aid the implementation of our approach. This way we were able to eliminate hardware difficulties faced by a physical system and also quickly debug any problems faced during the implementation process.

The RoboDK software was carefully chosen not because of the fact that it contains features for robot manipulator simulator but also because of the fully functional python application programming interface it provides. This API makes the development of an application very convenient which require external interfacing to the RoboDK software.

Some features of the software are as follows: -

- Easy to use.
- Fully functional Python API.
- Wide range of robot manipulator selection to choose from.
- Cross-platform.
- Inbuilt code editor.

Among others, RoboDK proved to be the perfect candidate for our idea and approach to pattern recognition.

## 3.3 Choice of programming language

Among all the programming languages available, we decided to implement our system in the python programming language due to various functionalities and tools it provides for computer vision and image processing application. We had almost no trouble setting up a system using this language. Moreover, the clean and short statements required to execute an instruction in the python language helps to keep things simple and intuitive in nature. Another reason for selecting the above mentioned programming language is that RoboDK provides an easy to use python application programming interface which can be used to make a manipulator accomplish different tasks.

## 3.4 Different phases of the approach

We present in this section the different steps the system goes through from the beginning to the termination of the program.

1. Initially, the robot simulation software takes into consideration the availability of a camera source which can be used as an input device. This input device will be used

   capture frames from the outside world. We have used the computer's inbuilt camera as the source of image frames taken at the default rate the webcam is configured at. The captured frames are then sent to the procedure which performs various operations on the image and extracts out useful information.

2. The image frames received by the main procedure are then processed and different operations are carried out on the raw image to make it suitable for feature extraction. Before extracting out features from the image, a small region of interest is segmented from the raw image on which the feature detection will be carried out. This step ensures that we only try to identify the patterns which are relevant to the application and eliminate those which are irrelevant. This step also makes the computation faster because we now have to process a smaller area of the image compared to the larger raw image.

3. In the third step the system performs a series of operations to make the selected region of interest suitable for extracting images. The following sequence of operations is carried out:-

   a. Convert the image to grayscale.
   b. Apply Gaussian Blur to remove noise from the image.
   c. Apply threshold function on the image to convert the image into a series of black and white pixels.

   The above sequence of operations performed on the selected region of the image makes the image free from any external disturbance and generates a relatively clean image for further feature detection.

4. This step corresponds to the feature detection mechanism implemented in eh resultant processed clean image obtained in the previous step. We try to detect the features by first finding the contours. The set of contours is then used to detect the convex hull. From the result obtained from the convex hull, we try detect the convexity defects.

The number of convexity defects is then used to determine the number of fingers shown in the selected area of the complete image.

The sequence of operations in this step is given below:-

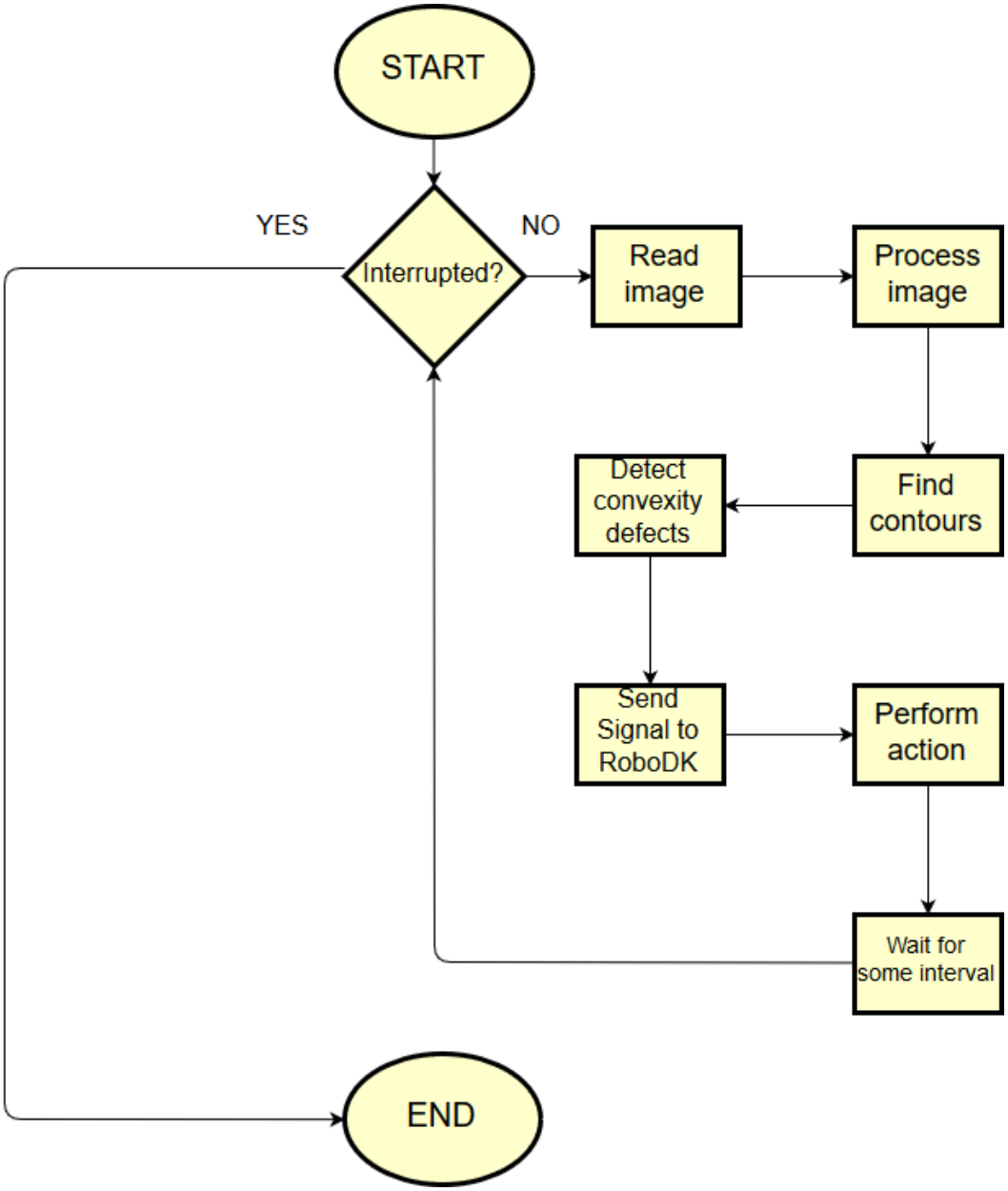
a. Find contours in the after applying threshold.
b. Find convexity defects.
c. The number of fingers is equal to the number of convexity defects plus one.

Using the above sequence of feature extraction, we can determine the number of fingers indirectly by calculating the number of convexity defects found in the image. The number of convexity defects is then passed to the next step where each signal of the is mapped to an action carried out by the robot manipulator.

**5.** The last step is to execute the respective actions mapped to each pattern. Each pattern of fingers, gesture, is mapped to an event where the robot draws a shape which consists of sides equivalent to the number of fingers detected. Here, the signal is sent to the RoboDK software in a separate thread which eliminates the blocking problem of the flow of the main program thread. Also, the concept of statistical mode is used to prevent accumulation of results. No reading is taken when the robot performs any action..

## 3.5 Flowchart

The flowchart for the above sequence of steps is given below.



## 3.6 Pseudocode for the proposed implementation

The steps listed out in the previous section can be aggregated in a concise sequence of steps as follows. It is assumed that all the requisite libraries are included and the RoboDK robot simulation software is running before the execution of the main program begins.

**Algorithm**

1. Initialize start and end timers
2. while (not interrupted) carry out:
    a. image = readImage ()
    b. image = filter(image)
    c. image = threshold(img)
    d. contours = findContours(image)
    e. defects = convexityDefects(contours)
    f. angle = acos(defects) // Applying cosine rule
    g. if(angle<=90) do:
        1. defect_count +=1
    h. sendSignal(defect_count) // Create new thread
    i. start = currentTime()
    j. waitToFinishAction ()
    k. stop = currentTime()
3. closeAllWindows ()
4. Terminate Execution // End

It can be clearly observed from the above pseudocode, the sequence of steps needed to carry out in order to obtain the correct results so that the correct action can be carried out by the robot manipulator.

The above approach detects the convexity defects and subsequently the number of fingers and correctly sends the signal to the robot manipulator and instruct it to draw a shape. The above approach can thus be incorporated into major areas of similar applications where a human can control a machine with his gestures by generating different patterns.

## 3.7 Summary

In this chapter we saw how the system goes through different phases to accomplish the task of recognizing the features and ultimately the finger count.

First, the phases of the system were explained which helped to get a detailed overview of the working of the system. Then the flowchart of the system was given to depict the control flow as a pictorial representation. Finally, the algorithm along with the pseudocode was provided which the core of the application.

So, in this chapter, we present the results and the provide detailed explanation of the system operation separated into different sections.

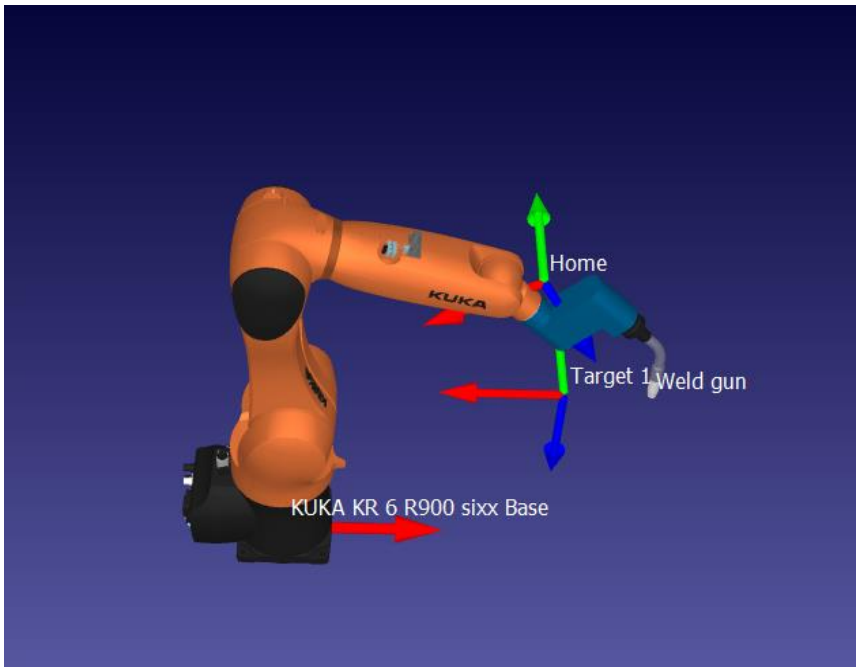
# Chapter 4

# Results and Discussion

---

## 4.1 Results

Differing from the already implemented models using microcontroller, the result of the bot is controlled by hand gestures (patterns). As explained in the implementation of the system, after the finger recognition the recognized pattern is fed to RoboDK. The robotic arm Fig(i)traces out the sides of polygon.



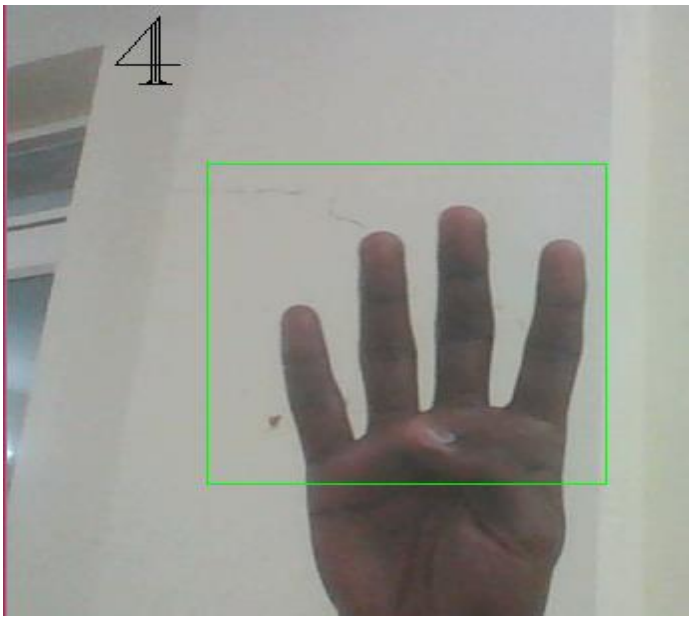

Fig(i)

Different cases taken into consideration are-

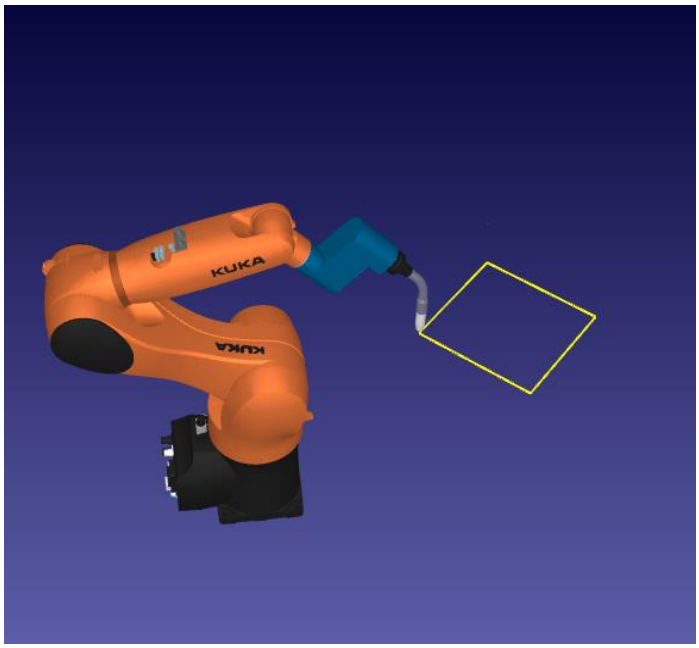
- 4-Finger Gesture
- 5-Finger Gesture
- Fogged 4-Finger Gesture
- Collective 5-Finger Gesture
- Oriented Finger Gesture

- **4-finger Gesture:**

A 4-finger input is given using the interface, as shown in Fig(ii). Using the input, the robotic arm draws the 4 sided polygon as shown in Fig(iii).



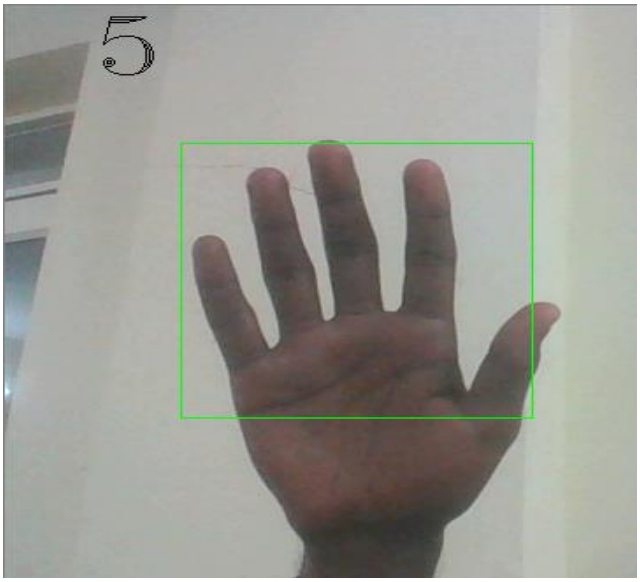

Fig(ii)



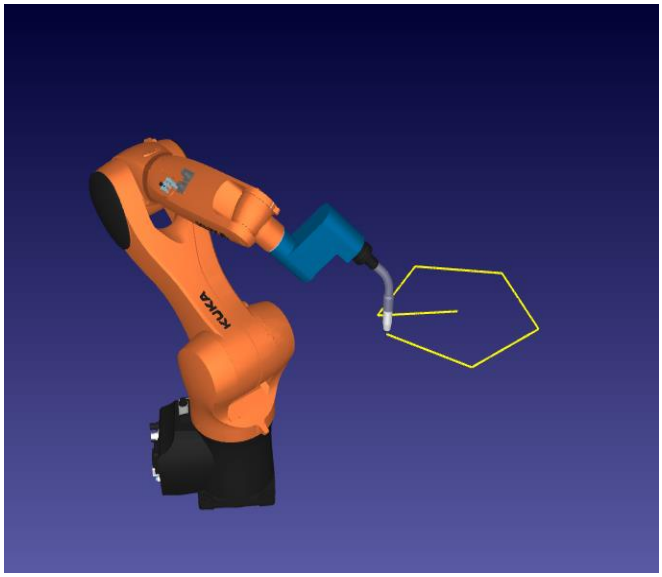

Fig(iii)

- **5-finger Gesture:**

A 5-finger input is given using the interface, as shown in Fig(iv). Using the input, the robotic arm draws the 4 sided polygon pentagon as shown in Fig(v).



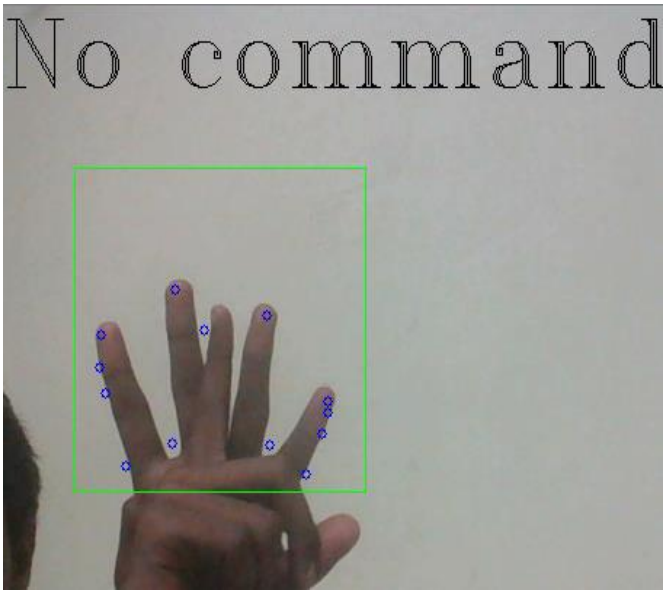

Fig(iv)




Fig(v)

- **Fogged 4-finger Gesture:**

A 4-finger clouded input is given using the interface, as shown in Fig(vi). No such pattern could be recognized thus giving no result.

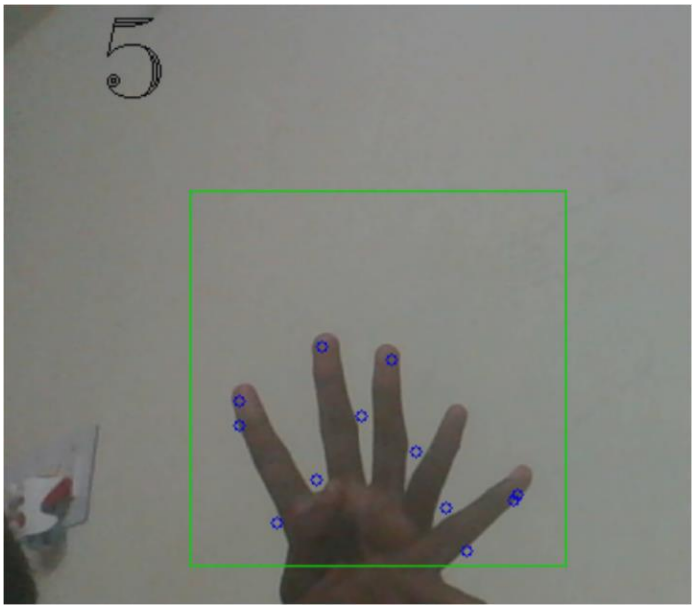



Fig(vi)

From the above picture we observe that though the convexity defects are identified by the system, no legitimate pattern is identified which is a correct result considering the fact that only a mathematical approach is being used.

- **Collective 5-finger Gesture:**

A 4-finger from one hand and 1-finger from another hand input is given using the interface, as shown in Fig(vii). Collectively the fingers are counted and making a 5 sided polygon pentagon.

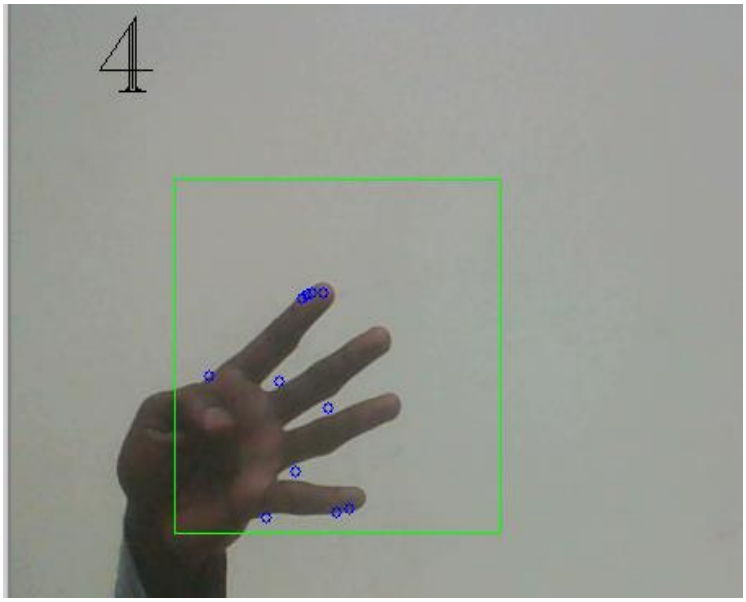



Fig(vii)

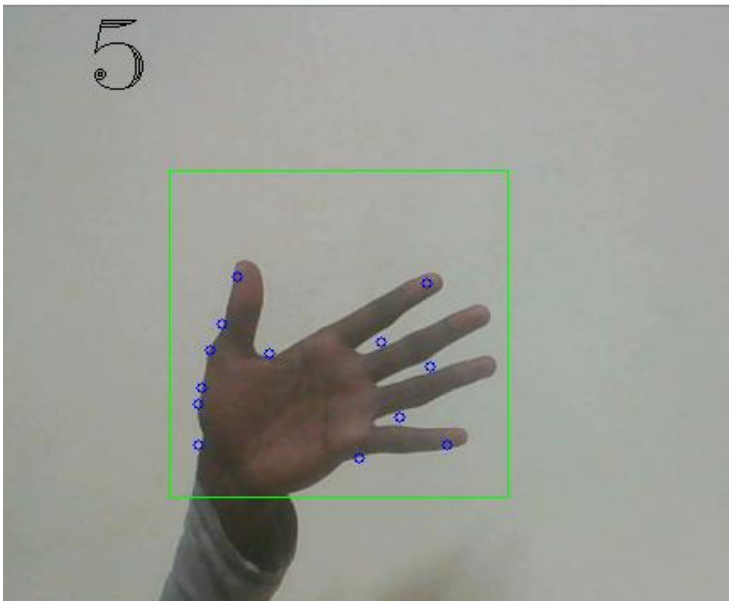
The above images show how the system correctly detects the number of fingers and counts them when the palm is oriented in a vertical position as normally a person would orient his or her hand towards the camera input source. In the following section, we show how the system is also correctly able to identify the number of fingers regardless of the orientation of the palm (fingers).

- **Oriented Finger Gesture:**

The finger gesture can also be given with certain orientation as an input to the interface. It will give the corresponding sided polygon as shown in Fig(viii) & Fig(ix).




Fig(viii)




Fig(ix)

## 4.2 Summary

In this section we saw the results obtained from the system and provided pictures for a proof of work and the steps carried out in the implementation chapter. Different results were provided where various cases are addressed and then a small explanation is provided to make the figures intuitive in nature and easy to understand. Lastly we provide some special cases where the system performs well and the

The different results above show that the system can take different pattern as input and give corresponding results. The whole process of input, detection and the display is very systematic and flexible. This makes the system more effective than the normal direction gesture controlled microcontroller bots.

# Chapter 5

# Conclusion

## 5.1 Summary

The traditional methods using microcontrollers are effective and brought about a great change in the automation industry. But to update the system with the latest technology we went through set of different implementations. Keeping different criteria's in mind we took up the gesture controlled robot using image processing.

We started with the selection of simulation software which could not only compensate the need of robot model but also provide us with the API that we thought of implementing in this model. RoboDK satisfied all our needs giving more flexibility of giving commands in Python programming language. Next step was image processing which included the image contouring to cropping the required finger gesture. This processing made the unrequired noise to be reduced thus making the system more efficient. The image detected was fed into result phase which commanded the robotic arm in RoboDK to draw the corresponding polygon.

The report gives a detailed information about the design and working of the system. It presents the simplicity and flexibility that the system has to provide with staircase wise implementation details. The results make our vision clearer.

The system is made by keeping future prospect in mind. The system is adaptive to further advancements making it effective and useful in different fields on science. Also, in the future we will work towards developing a more robust system with the application of different categories of neural networks and present a comparative analysis.

## 5.2 Cost Analysis

The robotic model being a simulation, there is no big project cost auditing required. But when implemented in a bigger project, the model may incur additional cost.

## 5.3 Challenges

The system encountered many challenges which were tackled using different approaches. Given below are the challenges with the corresponding solutions:

- Concurrent detection of multiple gestures led to accumulation of signals which hampered correct operation.
  Solution: The problem was resolved using statistical methods.
- The transmission of the message to the robot simulation software led to the blocking of the main program loop.
  Solution: This was resolved by applying multithreading.

## 5.4 Planning and Project Management

Following is the detailed project planning and management:

| Activity | Start Date | Number of weeks |
|---|---|---|
| Literature Review | 10-Feb | 1 |
| Finalizing problem definition | 18-Feb | 1 |
| Requirement gathering | 24-Feb | 1 |
| Implementation | 10-Mar | 2 |
| Result analysis | 24-Mar | 1 |
| Preparation of project report | 4-Apr | 1 |
| Preparation of project presentation | 4-Apr | 1 |

Fig(x)

The following figure shows the Gantt Chart which shows the time distribution of work.

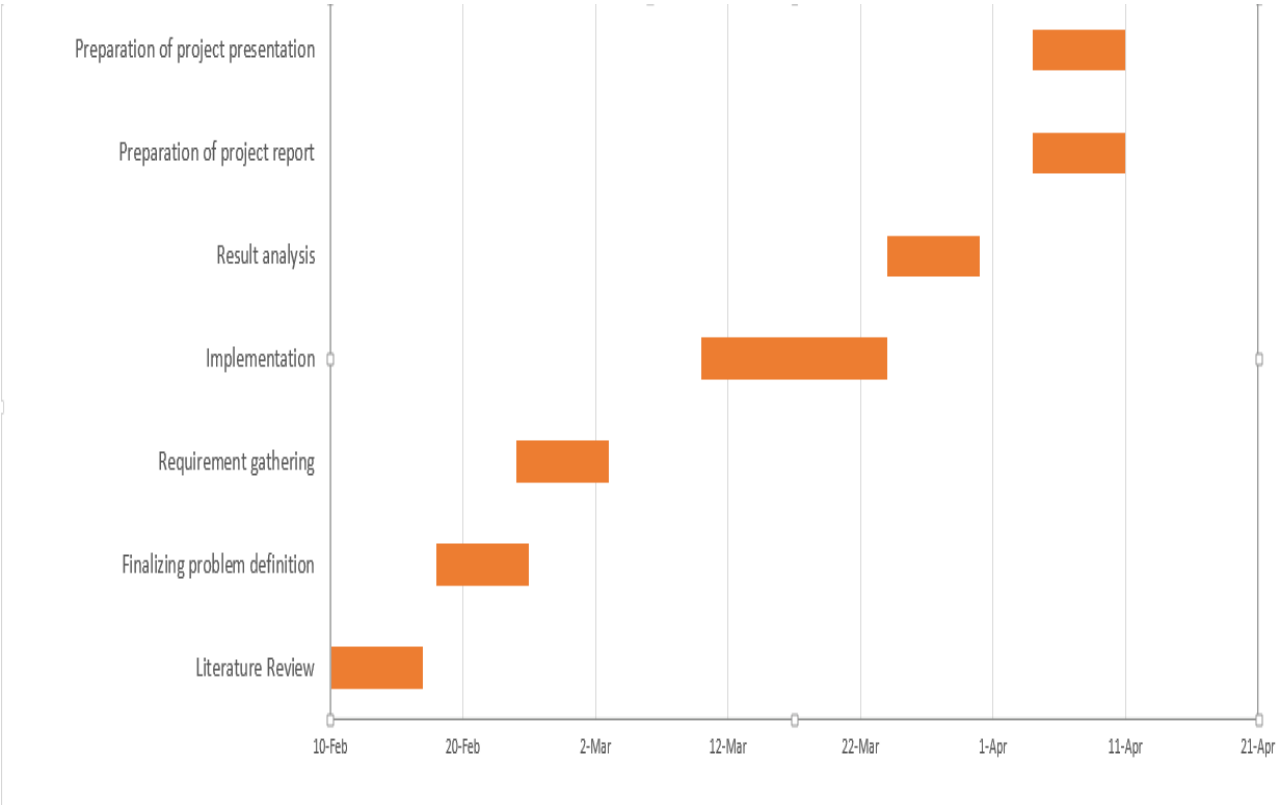



Fig (xi)

# Chapter 6

# References


1. SIGN LANGUAGE RECOGNITION USING 3D CONVOLUTIONAL NEURAL NETWORKS Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li

2. Hand Gesture Recognition with 3D Convolutional Neural Networks Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz NVIDIA, Santa Clara, California, USA

3. P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In CVPR, 2016.

4. RoboDK software https://www.robodk.com

5. RoboDK offline programming documentation https://robodk.com/offline-programming