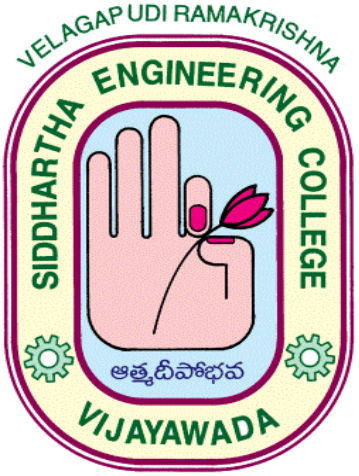
VELAGAPUDI RAMAKRISHNA SIDDHARTHA ENGINEERING COLLEGE,
VIJAYAWADA

A
PROJECT
REPORT
ON

HAND GESTURE RECOGNITION USING WEB CAMERA



Submitted To :
Impacteers Club

Submitted By :
Bonthu Kavya(198W1A0509)
Charan Abburi(198W1A0512)
Pranathi Dabbara(198w1A0514)
Ravulapalli Girija(198W1A0550)
Ganji Bhargava Sai(208W5A0501)
Batch Number : 03

Department Of Computer Science and Engineering
Velagapudi Ramakrishna Siddhartha Engineering College,Vijayawada

TABLE OF CONTENTS

List Of Images

Abstract

Gesture Recognition is the most favored and practicable solution to improve human interaction with computers and has been widely accepted in recent years thanks to its practice in many gaming devices such as ps4, Xbox, etc. as well as other devices such as laptops, smartphones, etc. The recognition of hand gestures is utilized in various applications such as accessibility support, crisis management, medicine etc. Hence, we came up with the project “Hand gesture Recognition” using Computer Vision and Deep Learning to guess the live hand gestures.

Introduction

Gesture:

A gesture is a development of a body part, which can be a hand, head, neckorsome other to express any emblematic portrayal of information.
This is a research on the development of interaction between human and computer using hand gestures.It is essentially a method for interfacing in a non-verbal way utilizing our body parts to convey the ideal message.
Using image processing techniques a vision based interaction can be established. Recognition of hand gestures using Ip. The inputs are taken from a web camera or any other camera. Hand motion acknowledgment is of incredible incentive in numerous apps, for example, acknowledgment of communication via gestures, enlarged reality (computer generated reality),gesture based communication mediators for the handicapped, and robotcontrol.

Gesture recognition features:

More accurate

High security/stability

Time saving to unlock a device

Some of the applications of hand gesture recognition are applied in the following sectors:

Automotive sector

Consumer electronics sector

Transit sector

Gaming sector

To unlock smartphones Defence

Home automation

Automated sign language translation

Motion acknowledgment can be led with procedures from PC vision and picture preparing.

The Gestures can be classified into two sub -classes

- StaticGestures
- DynamicGestures

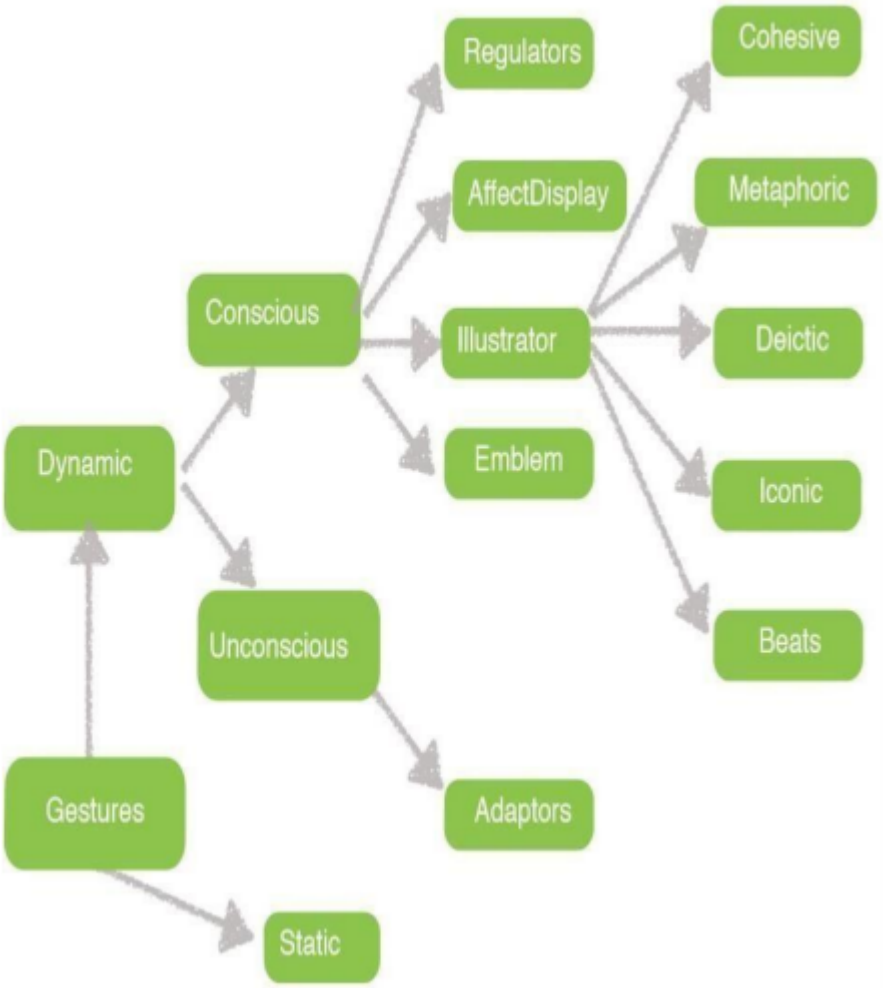


Figure 1. Gestures Classification

Problem Statement

"Hand Gesture Recognition Using Camera " is based on the concept of Image processing. In recent years there is a lot of research on gesture recognition using kinect sensor on using HD camera but camera and kinect sensors are more costly. This paper is focused on reducing cost and improving robustness of the proposed system using simple web camera.

Methodology

The system developed using computer vision and deep learning. This model takes in a video feed and gives out the results of what gesture is shown by the person in the video. This process is achieved by using OpenCV for video handling, deep learning for recognition and also math packages in python.

Architecture

The model that has been built in this project takes in a video input and gives the recognised output.

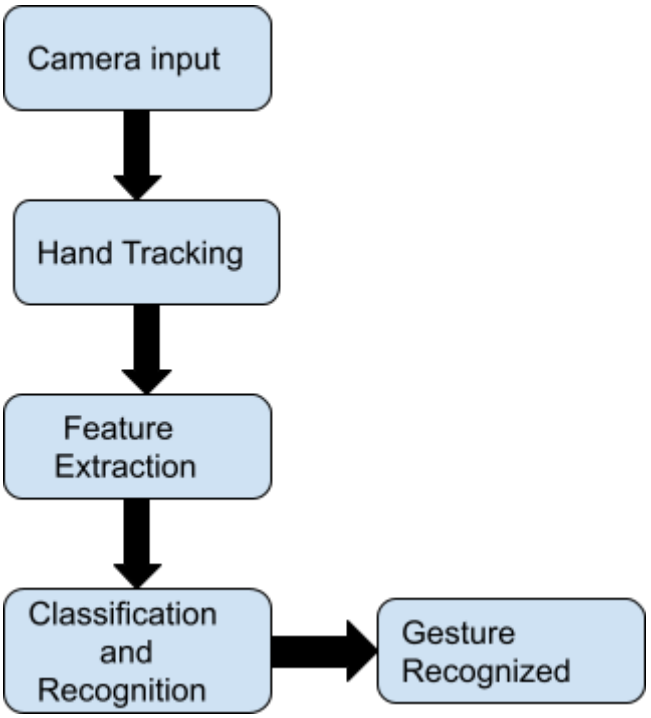


Figure 2. Architecture of the Model

Software and Hardware used:

Software details:

Programming Language: Python

Machine Learning packages:

1. OpenCV
2. Numpy

Other packages: Math

Hardware details:

The experiment was performed on Lenovo S145-151WL having specifications:

- RAM - 8.00 GB
- Windows 10
- Intel(R) Core(TM)i5-8265U CPU @ 1.60GHz 1.80GHz

Use of Camera:

The essential capacity of this framework is to perceive hand signals. Here, a motion picture is taken utilizing the camera, the picture will be handled by the model and a yield will be given on the showcase screen.

Technical Description

Most gesture recognition methods usually contain three major stages. The first stage is object detection. The target of this stage is to detect hand objects in the digital images or videos. The second stage is object recognition. The detected hand objects are recognized to identify the gestures. The third stage is to analyze sequential gestures to identify users' instructions or behaviors.

Source Code

```
import cv2
import numpy as np
import math
cap = cv2.VideoCapture(0)

while(1):

    try:  #an error comes if it does not find anything in
window as it cannot find contour of max area
          #therefore this try error statement

        ret, frame = cap.read()
        frame=cv2.flip(frame,1)
        kernel = np.ones((3,3),np.uint8)

        #define region of interest
        roi=frame[100:300, 100:300]



cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
        hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)



    # define range of skin color in HSV
        lower_skin = np.array([0,20,70], dtype=np.uint8)
        upper_skin = np.array([20,255,255],
dtype=np.uint8)

     #extract skin color image
        mask = cv2.inRange(hsv, lower_skin, upper_skin)



    #extrapolate the hand to fill dark spots within
        mask = cv2.dilate(mask,kernel,iterations = 4)

    #blur the image
        mask = cv2.GaussianBlur(mask,(5,5),100)
```

```
    #find contours
        _,contours,hierarchy=
cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPL
E)

   #find contour of max area(hand)
        cnt = max(contours, key = lambda x:
cv2.contourArea(x))

    #approx the contour a little
        epsilon = 0.0005*cv2.arcLength(cnt,True)
        approx= cv2.approxPolyDP(cnt,epsilon,True)


    #make convex hull around hand
        hull = cv2.convexHull(cnt)

     #define area of hull and area of hand
        areahull = cv2.contourArea(hull)
        areacnt = cv2.contourArea(cnt)

    #find the percentage of area not covered by hand in
convex hull
        arearatio=((areahull-areacnt)/areacnt)*100

     #find the defects in convex hull with respect to hand
        hull = cv2.convexHull(approx, returnPoints=False)
        defects = cv2.convexityDefects(approx, hull)

    # l = no. of defects
        l=0

    #code for finding no. of defects due to fingers
        for i in range(defects.shape[0]):
            s,e,f,d = defects[i,0]
            start = tuple(approx[s][0])
            end = tuple(approx[e][0])
            far = tuple(approx[f][0])
            pt= (100,180)


            # find length of all sides of triangle
```

```
            a = math.sqrt((end[0] - start[0])**2 + (end[1]
- start[1])**2)
            b = math.sqrt((far[0] - start[0])**2 + (far[1]
- start[1])**2)
            c = math.sqrt((end[0] - far[0])**2 + (end[1] -
far[1])**2)
            s = (a+b+c)/2
            ar = math.sqrt(s*(s-a)*(s-b)*(s-c))

            #distance between point and convex hull
            d=(2*ar)/a

            # apply cosine rule here
            angle = math.acos((b**2 + c**2 -
a**2)/(2*b*c)) * 57


            # ignore angles > 90 and ignore points very
close to convex hull(they generally come due to noise)
            if angle <= 90 and d>30:
                l += 1
                cv2.circle(roi, far, 3, [255,0,0], -1)

            #draw lines around hand
            cv2.line(roi,start, end, [0,255,0], 2)


        l+=1

        #print corresponding gestures which are in their
ranges
        font = cv2.FONT_HERSHEY_SIMPLEX
        if l==1:
            if areacnt<2000:
                cv2.putText(frame,'Put hand in the
box',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
            else:
                if arearatio<12:
                    cv2.putText(frame,'0',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)
                elif arearatio<17.5:
                    cv2.putText(frame,'Best of
luck',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

```
                else:
                    cv2.putText(frame,'1',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)

        elif l==2:
            cv2.putText(frame,'2',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)

        elif l==3:

              if arearatio<27:
                    cv2.putText(frame,'3',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)
              else:
                    cv2.putText(frame,'ok',(0,50), font,
2, (0,0,255), 3, cv2.LINE_AA)

        elif l==4:
            cv2.putText(frame,'4',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)

        elif l==5:
            cv2.putText(frame,'5',(0,50), font, 2,
(0,0,255), 3, cv2.LINE_AA)

        elif l==6:
            cv2.putText(frame,'reposition',(0,50), font,
2, (0,0,255), 3, cv2.LINE_AA)

        else :
            cv2.putText(frame,'reposition',(10,50), font,
2, (0,0,255), 3, cv2.LINE_AA)
        #show the windows
        cv2.imshow('mask',mask)
        cv2.imshow('frame',frame)
    except:
        pass


    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()
```

```
cap.release()
```

Process Diagram

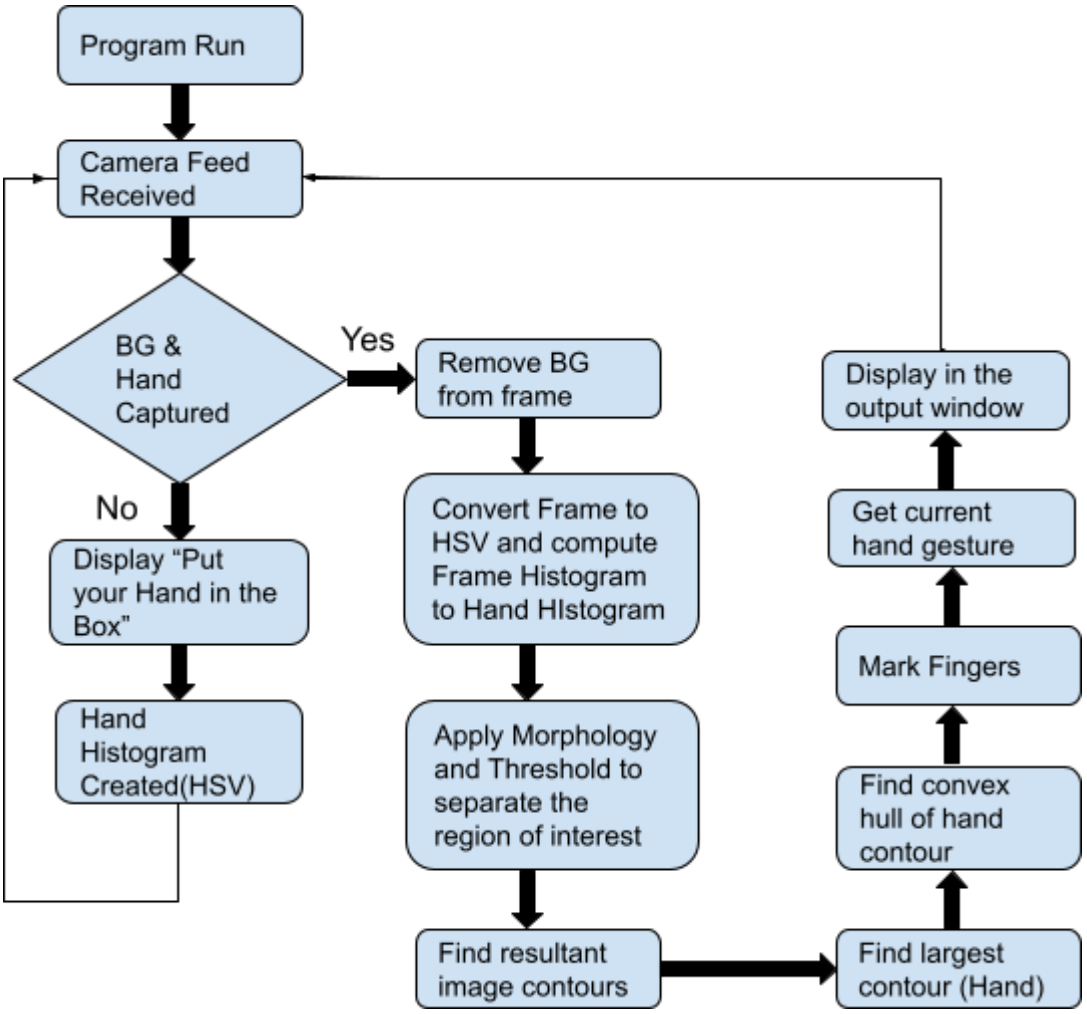


Figure 3. Detailed process flow of proposed algorithm

Results

Below are the images of the output given by the model.

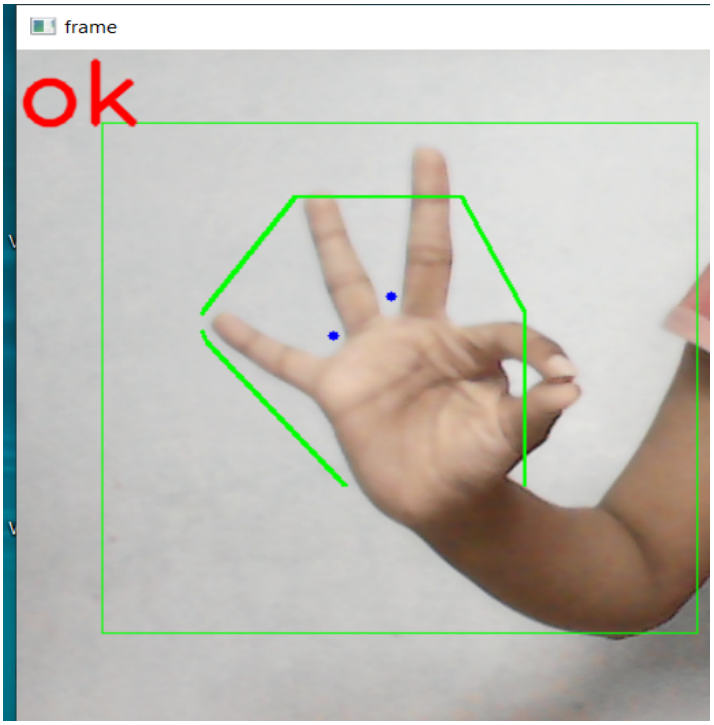


Figure 4. Gesture “Ok”

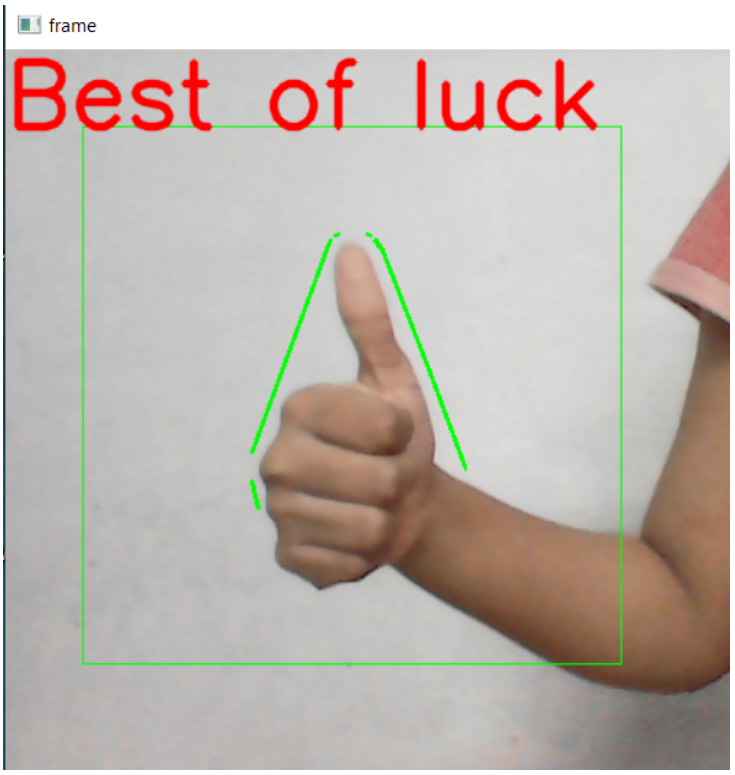


Figure 5: Gesture "Best of Luck"

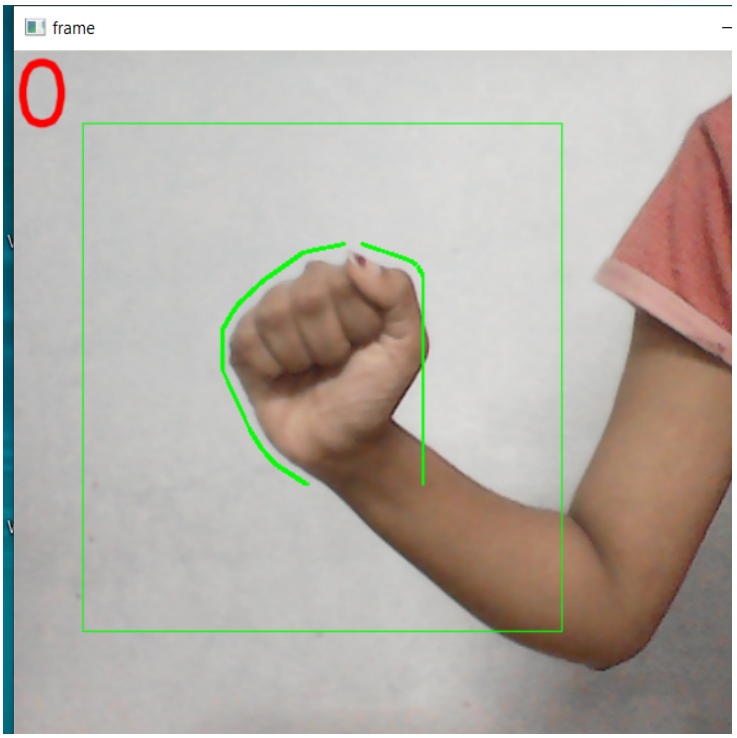


Figure 6. Gesture "0"

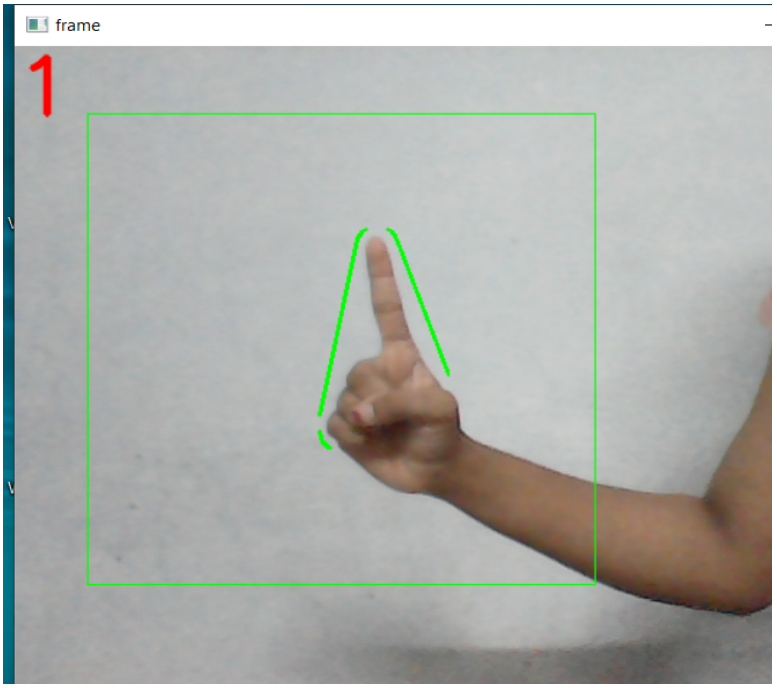


Figure 7. Gesture “1”

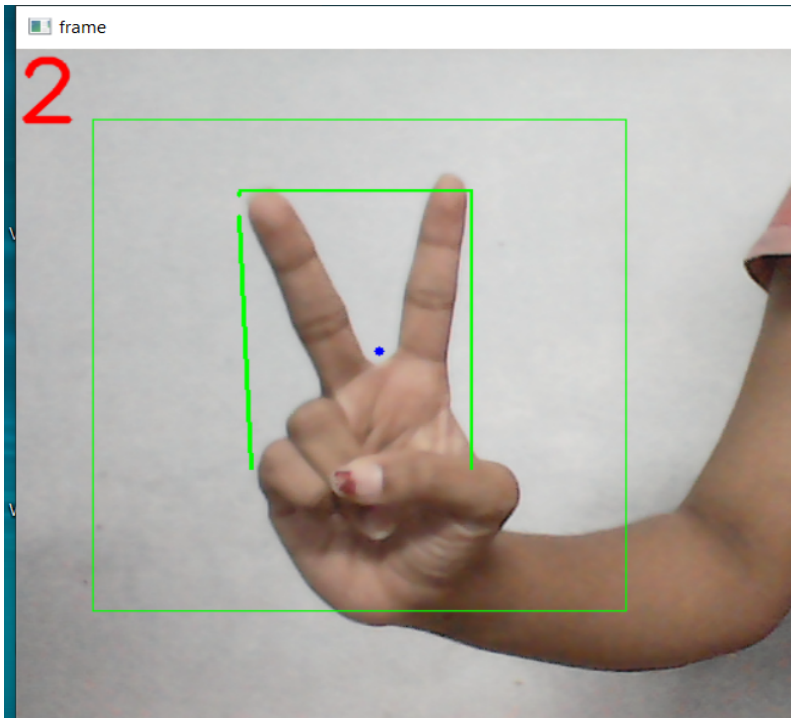


Figure 8. Gesture “2”

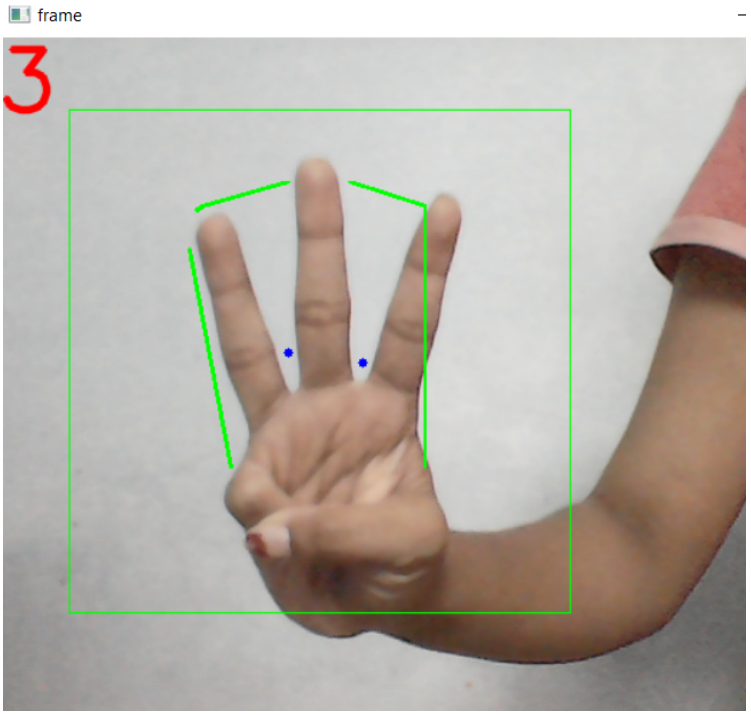


Figure 9. Gesture “3”

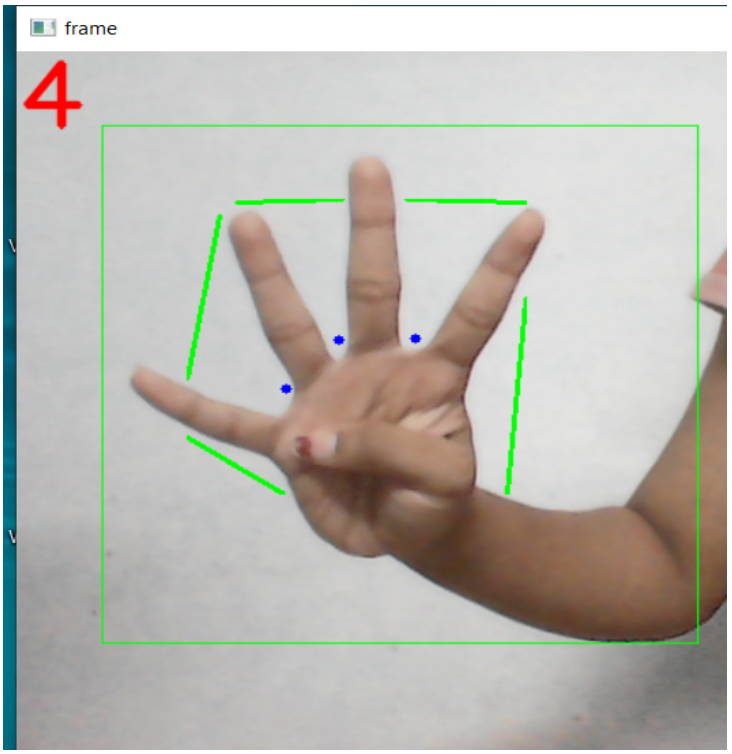


Figure 10. Gesture “4”

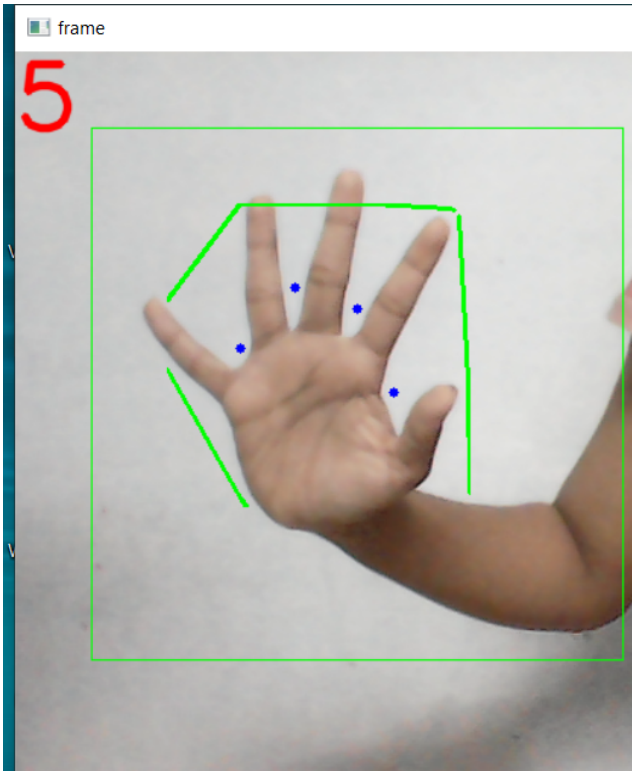


Figure 11. Gesture “5”