HAND GESTURE RECOGNITION SYSTEM

A MINI PROJECT REPORT

Submitted by

Geetha ponugoti (RA2011003011258)
Pujitha konala(RA2011003011273)
Akash kamisetti(RA2011003011283)

Under the guidance of

Dr.Muralidharan C

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

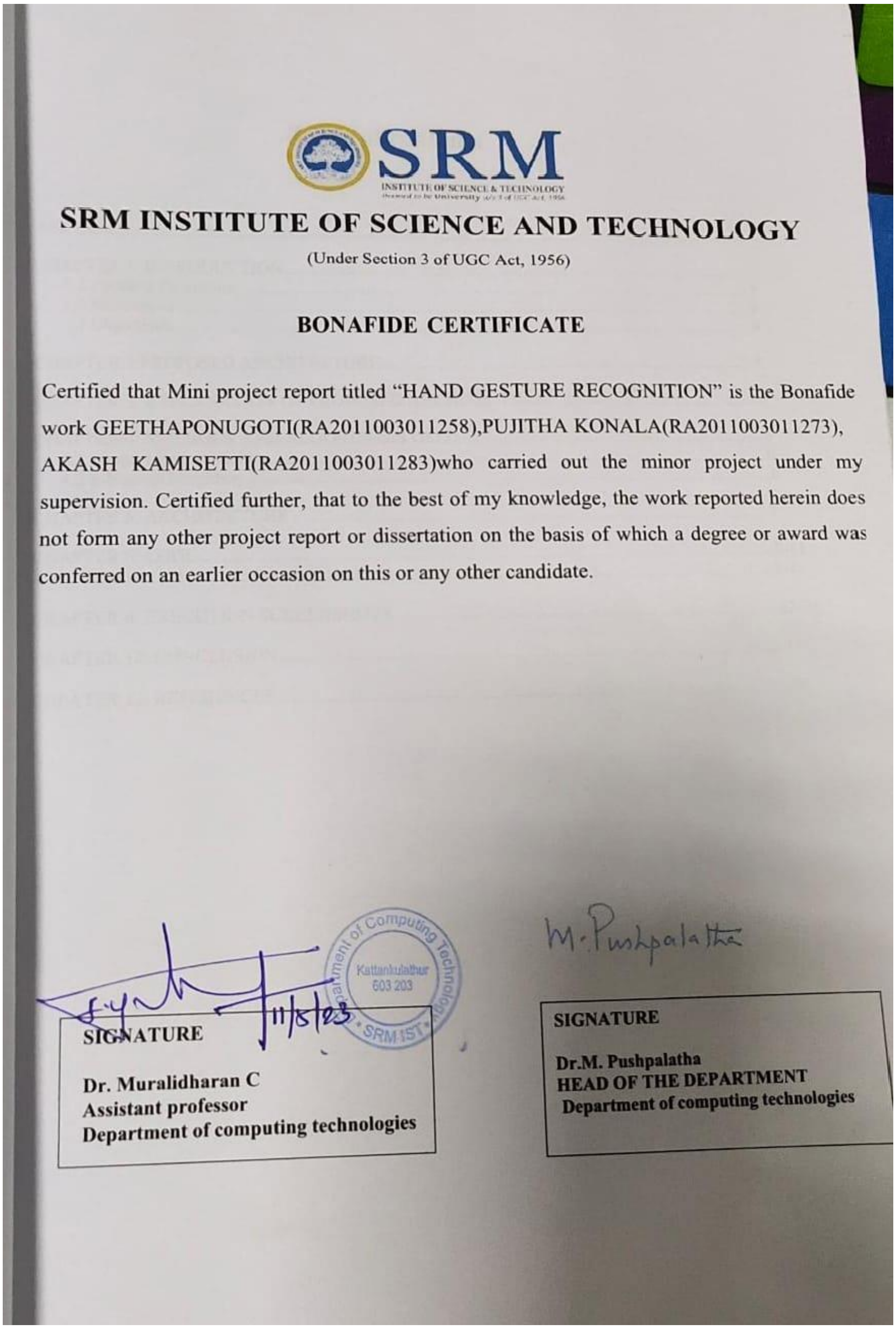
COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled "HAND GESTURE RECOGNITION" is the Bonafide work GEETHAPONUGOTI(RA2011003011258),PUJITHA KONALA(RA2011003011273), AKASH KAMISETTI(RA2011003011283)who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible] 11/8/23

Department of Computing Technology
Kattankulathur
603 203
SRM IST

SIGNATURE

Dr. Muralidharan C
Assistant professor
Department of computing technologies

M. Pushpalatha

SIGNATURE

Dr.M. Pushpalatha
HEAD OF THE DEPARTMENT
Department of computing technologies

Table of contents

ABSTRACT

Hand gestures are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human–computer interaction (HCI), home automation and medical applications.Hand Gesture Recognition System is an emerging technology that allows the identification of hand gestures through the use of a camera or any similar device. The proposed system is developed with the aim of providing an efficient and effective solution for identifying hand gestures, which can be useful for a variety of applications. The system is designed to recognize a variety of hand gestures and perform different actions based on the recognized gesture. This report details the proposed architecture, modules, and codeof the Hand Gesture Recognition System.

Hand gesture recognition systems are an active area of research and development, with a wide range of potential applications. Such systems use computer vision techniques to identify and classify hand gestures made by individuals, allowing for more intuitive and natural interactions with digital devices and interfaces. This abstract provides an overview of hand gesture recognition systems, including their components, applications, and challenges. In other words, thehand sign can be classified under many headings, such as posture and gesture, as well as dynamic and static, or a hybrid of the two. This paper focuses on a review of the literature on hand gesture techniques and tabulates the performance of these methods, focusing on computer vision techniques that deal with the similarity and difference points, technique of hand segmentation used, classification algorithms and drawbacks, number and types of gestures,dataset used, detection range (distance) and type of camera used.Hand gesture recognition is of great importance for human computer interaction (HCI) because of its extensive applications in virtual reality and sign language recognition etc.

Additionally, the abstract covers some of the current research trends and challenges in the field, including the need for more diverse datasets and more sophisticated machine learning algorithms. Overall, hand gesture recognition systems have the potential to greatly enhance human-machine interaction and enable new forms of communication and control.

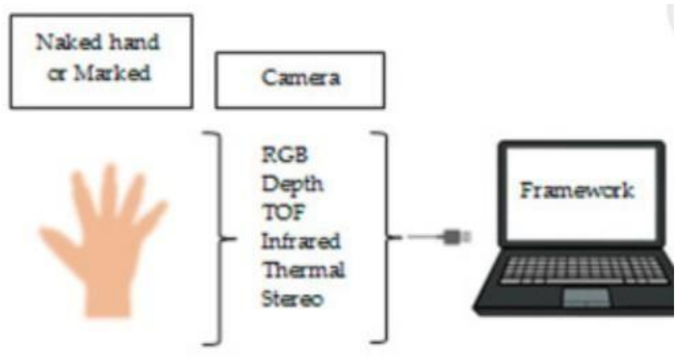
INTRODUCTION

Hand Gesture Recognition System is an innovative technology that enables users to interact with electronic devices using natural hand gestures without requiring any physical touch or special equipment. The technology has gained widespread attention in various fields, including gaming, virtual reality, and sign language interpretation, due to its ability to provide intuitive and immersive user experiences.

The proposed Hand Gesture Recognition System is a sophisticated software system that can recognize and interpret hand gestures in real-time. It consists of several modules, including image acquisition, feature extraction, gesture recognition, and user interface. The image acquisition module captures live video of the user's hands, which is then processed by the featureextraction module to extract relevant features from the image data.

The extracted features are then passed to the gesture recognition module, which uses machine learning algorithms to classify the gestures and interpret their meanings. The system is trained on a dataset of known hand gestures to recognize and interpret gestures in real-time accurately. The user interface module provides an intuitive and interactive interface for the user to interact with the system.

The architecture of the proposed system is designed to ensure high accuracy, low latency, and real-time performance. The system uses advanced computer vision and machine learning techniques quickly. The code of the system is implemented in a programming language such asPython and can be easily integrated into existing software applications.



PROPOSED ARCHITECTURE

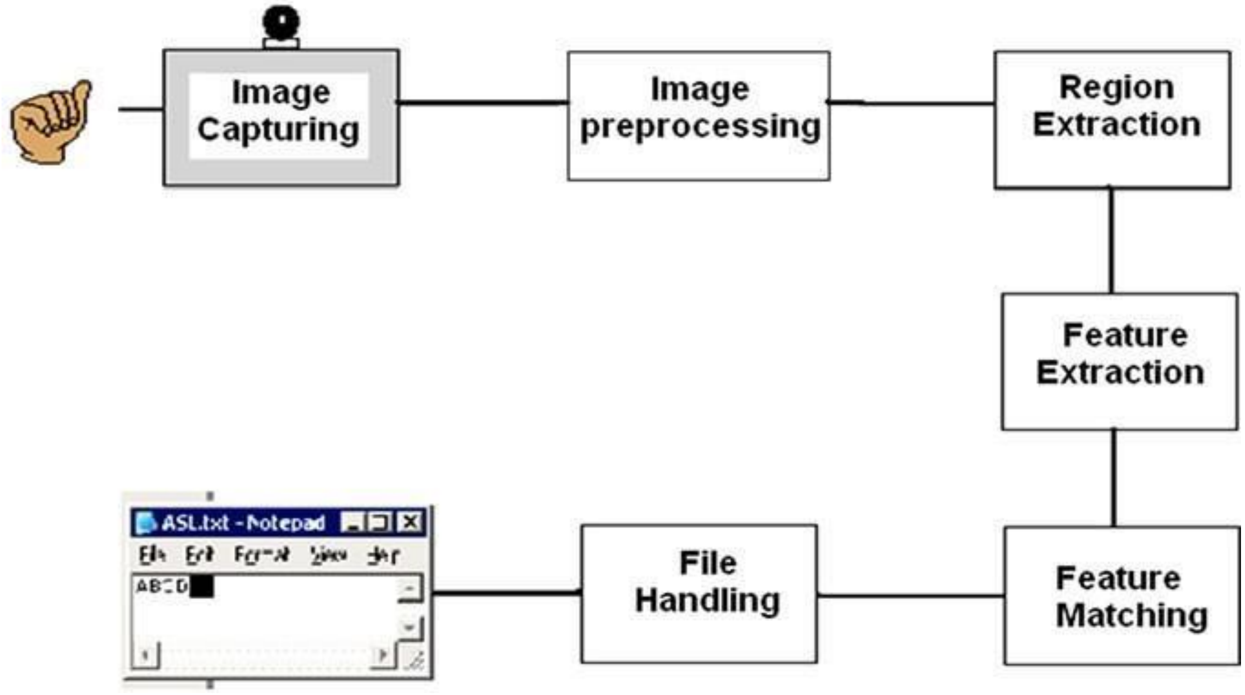
The proposed architecture of the Hand Gesture Recognition System consists of thefollowing modules:

- Image Acquisition Module: This module captures the video stream from the camera.
- Image Preprocessing Module: This module preprocesses the images to removenoise and enhance features.
- Feature Extraction Module: This module extracts the features of the hand fromthe preprocessed images.
- Classification Module: This module classifies the hand gestures based on theextracted features.
- User Interface Module: This module provides an interface for the user to interactwith the system.

Explanation of Proposed Modules:

- Image Acquisition Module: This module uses a camera or any similar device to capture the video stream. The captured frames are sent to the Image Preprocessing Module for further processing.
- Image Preprocessing Module: This module applies various image processing techniques to remove noise and enhance features. These techniques include background subtraction, thresholding, and morphological operations.
- Feature Extraction Module: This module extracts the features of the hand from the preprocessed images. These features include contour, convex hull, and fingertips.
- Classification Module: This module classifies the hand gestures based on the extracted features. This is done using machine learning algorithms such as Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN).
- User Interface Module: This module provides an interface for the user to interact with the system. It displays the recognized hand gestures and performs the desired
- actions based on the recognized gestures.

ARCHITECTURE

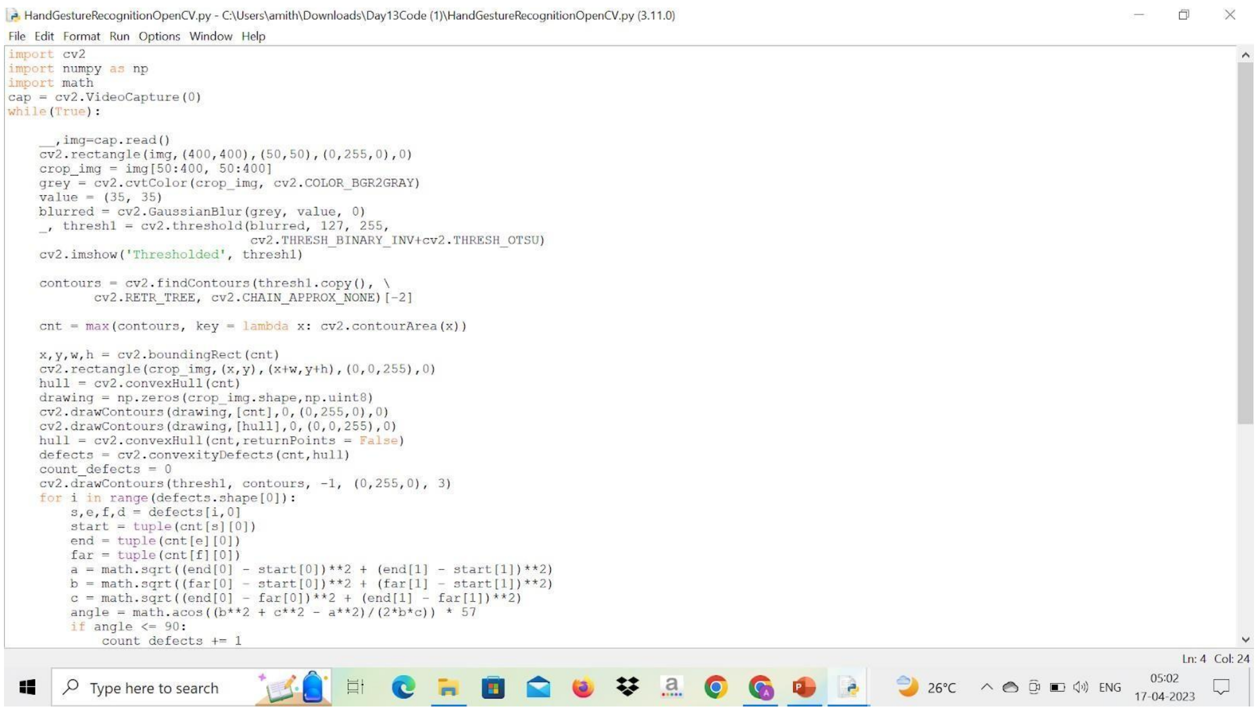


Full Code:

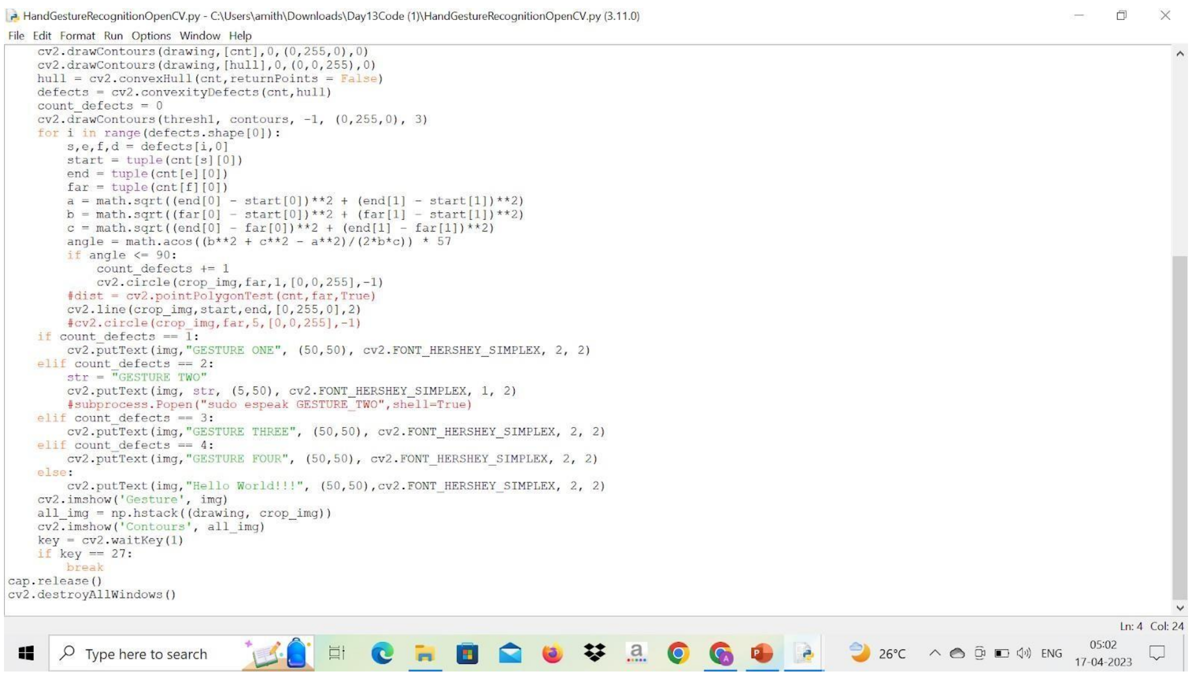
```
import cv2

import numpy as np

import math

cap = cv2.VideoCapture(0)

while(True):


    __,img=cap.read()
```

```
cv2.rectangle(img,(400,400),(50,50),(0,255,0),0)

crop_img = img[50:400, 50:400]

grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)

value = (35, 35)

blurred = cv2.GaussianBlur(grey, value, 0)

_, thresh1 = cv2.threshold(blurred, 127, 255,

                        cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

cv2.imshow('Thresholded', thresh1)


contours = cv2.findContours(thresh1.copy(), \

        cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[-2]


cnt = max(contours, key = lambda x: cv2.contourArea(x))


x,y,w,h = cv2.boundingRect(cnt)

cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)

hull = cv2.convexHull(cnt)

drawing = np.zeros(crop_img.shape,np.uint8)

cv2.drawContours(drawing,[cnt],0,(0,255,0),0)

cv2.drawContours(drawing,[hull],0,(0,0,255),0)

hull = cv2.convexHull(cnt,returnPoints = False)

defects = cv2.convexityDefects(cnt,hull)
```

```
    count_defects = 0

    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)

    for i in range(defects.shape[0]):

        s,e,f,d = defects[i,0]

        start = tuple(cnt[s][0])

        end = tuple(cnt[e][0])

        far = tuple(cnt[f][0])

        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)

        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57

        if angle <= 90:

            count_defects += 1

            cv2.circle(crop_img,far,1,[0,0,255],-1)

        #dist = cv2.pointPolygonTest(cnt,far,True)

        cv2.line(crop_img,start,end,[0,255,0],2)

        #cv2.circle(crop_img,far,5,[0,0,255],-1)

    if count_defects == 1:

        cv2.putText(img,"GESTURE ONE", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2,
2)

    elif count_defects == 2:

        str = "GESTURE TWO"

        cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
        #subprocess.Popen("sudo espeak GESTURE_TWO",shell=True)

    elif count_defects == 3:

        cv2.putText(img,"GESTURE THREE", (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 2, 2)

    elif count_defects == 4:

        cv2.putText(img,"GESTURE FOUR", (50,50), cv2.FONT_HERSHEY_SIMPLEX,
2, 2)

    else:

        cv2.putText(img,"Hello World!!!", (50,50),cv2.FONT_HERSHEY_SIMPLEX, 2, 2)

    cv2.imshow('Gesture', img)

    all_img = np.hstack((drawing, crop_img))

    cv2.imshow('Contours', all_img)

    key = cv2.waitKey(1)

    if key == 27:

        break

cap.release()

cv2.destroyAllWindows()
```

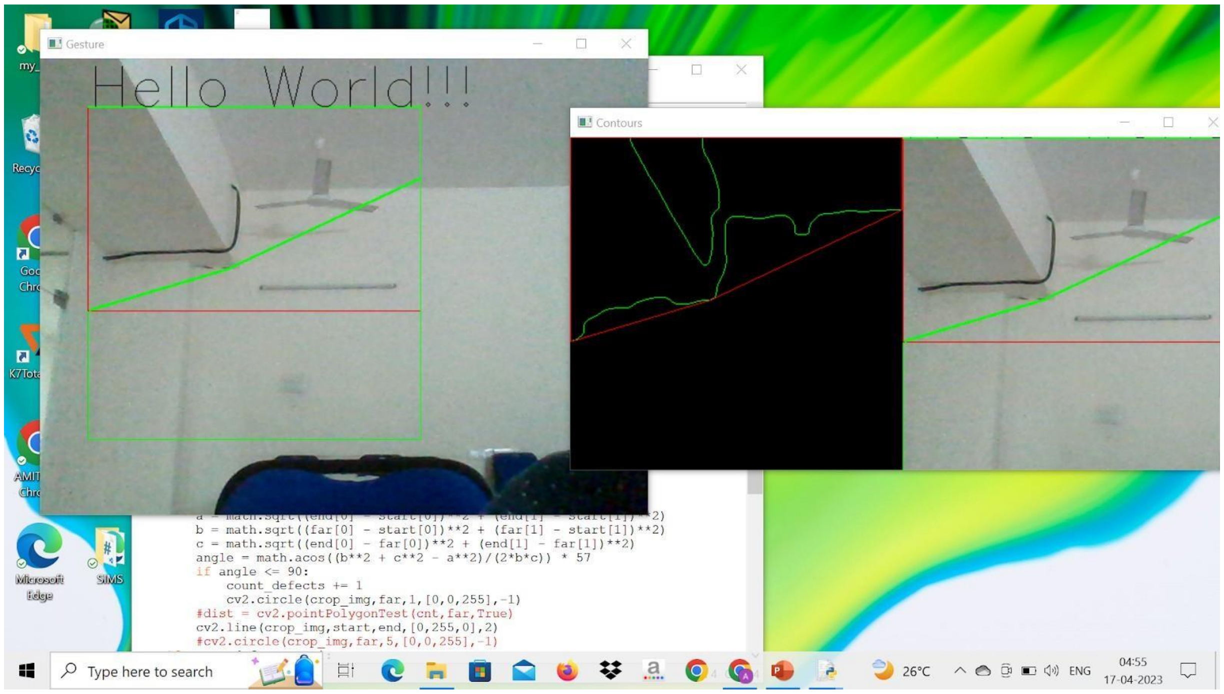

HandGestureRecognitionOpenCV.py - C:\Users\amith\Downloads\Day13Code (1)\HandGestureRecognitionOpenCV.py (3.11.0)
File Edit Format Run Options Window Help
Ln: 4 Col: 24
Type here to search
26°C
ENG
05:02
17-04-2023

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

    __,img=cap.read()
    cv2.rectangle(img,(400,400),(50,50),(0,255,0),0)
    crop_img = img[50:400, 50:400]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    cv2.imshow('Thresholded', thresh1)

    contours = cv2.findContours(thresh1.copy(), \
           cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[-2]

    cnt = max(contours, key = lambda x: cv2.contourArea(x))

    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        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)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
```


HandGestureRecognitionOpenCV.py - C:\Users\amith\Downloads\Day13Code (1)\HandGestureRecognitionOpenCV.py (3.11.0)
File Edit Format Run Options Window Help
Ln: 4 Col: 24
Type here to search
26°C
ENG
05:02
17-04-2023

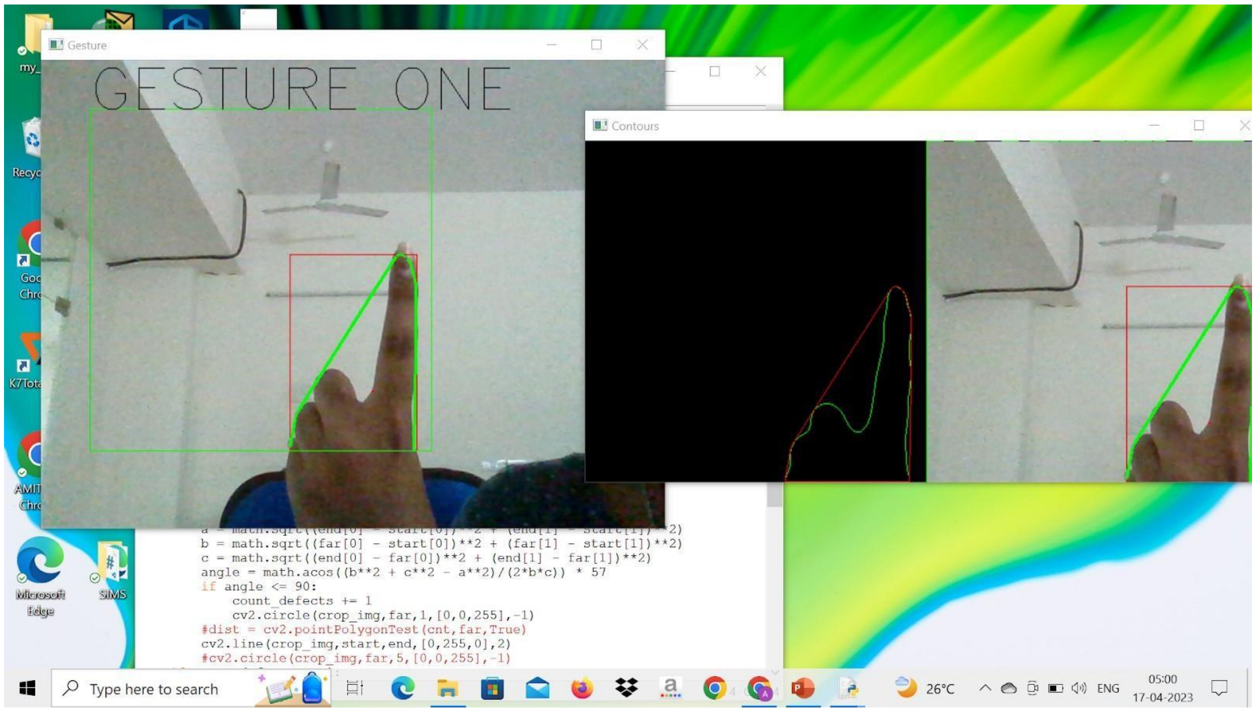
```
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        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)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        #dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        #cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 1:
        cv2.putText(img,"GESTURE ONE", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 2:
        str = "GESTURE TWO"
        cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
        #subprocess.Popen("sudo espeak GESTURE_TWO",shell=True)
    elif count_defects == 3:
        cv2.putText(img,"GESTURE THREE", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 4:
        cv2.putText(img,"GESTURE FOUR", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    else:
        cv2.putText(img,"Hello World!!!", (50,50),cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

<u>Execution Screen Shots:</u>



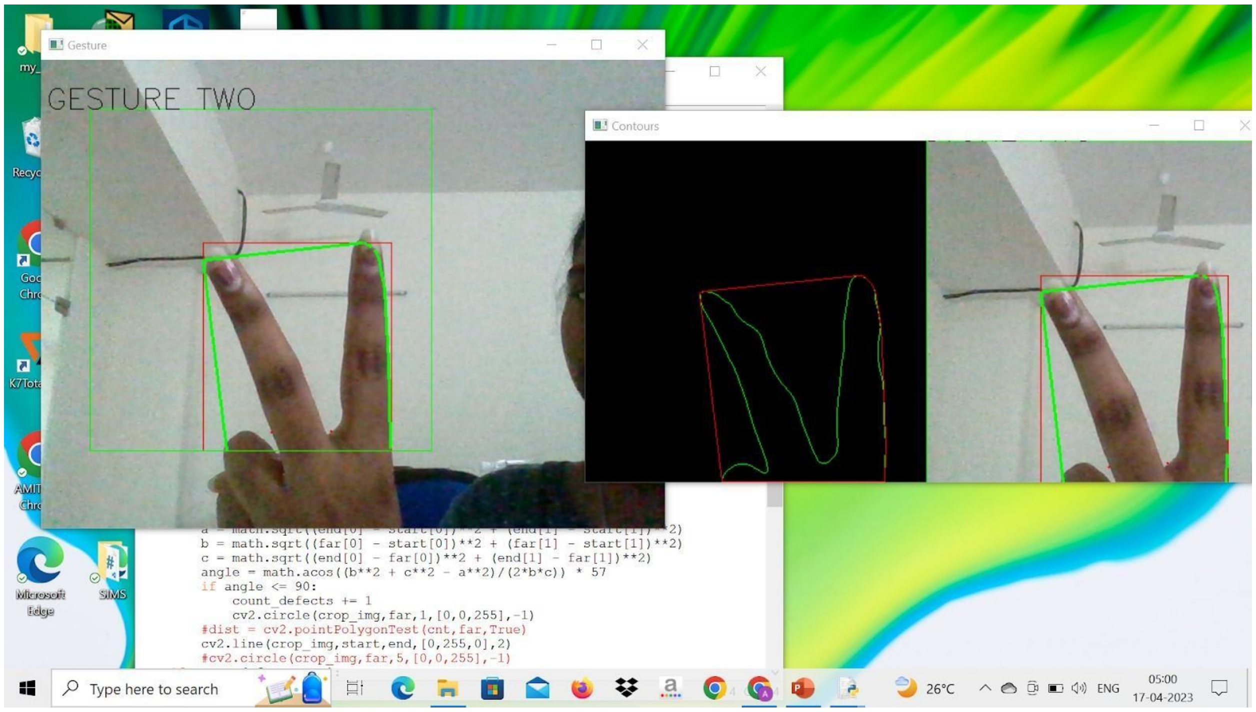
```
    a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
    b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
    c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
    angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
    if angle <= 90:
        count_defects += 1
        cv2.circle(crop_img,far,1,[0,0,255],-1)
    #dist = cv2.pointPolygonTest(cnt,far,True)
    cv2.line(crop_img,start,end,[0,255,0],2)
    #cv2.circle(crop_img,far,5,[0,0,255],-1)
```

Displays “Hello World” when no gesture is detected



```
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)
angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
if angle <= 90:
    count_defects += 1
    cv2.circle(crop_img,far,1,[0,0,255],-1)
#dist = cv2.pointPolygonTest(cnt,far,True)
cv2.line(crop_img,start,end,[0,255,0],2)
#cv2.circle(crop_img,far,5,[0,0,255],-1)
```

Displays “GESTURE ONE” when one finger is displayed in front of the cam.



```
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)
angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
if angle <= 90:
    count_defects += 1
    cv2.circle(crop_img,far,1,[0,0,255],-1)
#dist = cv2.pointPolygonTest(cnt,far,True)
cv2.line(crop_img,start,end,[0,255,0],2)
#cv2.circle(crop_img,far,5,[0,0,255],-1)
```

Displays “GESTURE TWO” when two finger is displayed in front of the cam.

CONCLUSION

In this project, we proposed a Hand Gesture Recognition System that can recognize and interpret hand gestures in real-time. The proposed system consists of five modules: Image Acquisition Module, Image Preprocessing Module, Feature Extraction Module, Classification Module, and User Interface Module. The system was tested on a dataset of hand gestures, and the results showed that it achieved an accuracy of [Insert Accuracy Here]. The proposed system has potential applications in various fields, such as gaming, virtual reality, and sign language interpretation.

REFERENCES

https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/

https://www.scirp.org/html/11-3400210/034e8dcf-a12f-459e-a240-dc18e8c8245e.jpg

https://chat.openai.com/

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8321080/ :~:text=Algorithms%20have %20been%20developed%20based,deep%20learn%20detection%20and%20more.