MAULANA AZAD

NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL, MADHYA PRADESH,462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Handy Numbers : Finger Counting

Minor Project Report

VI Semester

Submitted by:

Aryav Dubey	**181112003**
Shivansh Pandey	**181112009**
Shubham Yadav	**181112029**
Sumit Rathore	**181112030**

Under the Guidance of:

Dr. Vaibhav Soni

Department of Computer Science and Engineering

Session (2020 – 21)

MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project report carried out on "**Handy Numbers : Finger Counting"** by the 3rd year students:

Aryav Dubey	**181112003**
Shivansh Pandey	**181112009**
Shubham Yadav	**181112029**
Sumit Rathore	**181112030**

Have successfully completed their project in partial fulfilment of their Degree in Bachelor of Technology in Computer Science and Engineering.

Dr. Vaibhav Soni
(Assistant Professor, CSE MANIT)

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL, BHARAT,462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, hereby declare that the following report which is being presented in the Minor Project Documentation Entitled as "**HANDY NUMBERS : FINGER COUNTING**" is an authentic documentation of our own original work and to the best of our knowledge. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to the research by others, with whom we have worked at Maulana Azad National Institute of Technology, Bhopal or elsewhere, is explicitly acknowledged in thereport.

We, hereby declare that the facts mentioned above are true to the best of our knowledge. In case of any unlikely discrepancy that may possibly occur, we will take theresponsibility**.**

Aryav Dubey	**181112003**
Shivansh Pandey	**181112009**
Shubham Yadav	**181112029**
Sumit Rathore	**181112030**

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide and coordinator thank **Dr. Vaibhav Soni** (Assistant Professor, CSE MANIT) for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully.

It is imperative for us to mention the fact that the report of minor project could not have been accomplished without the periodic suggestions and advice of our project guide thank **Dr. Vaibhav Soni** (Assistant Professor, CSE MANIT) and project coordinators **Dr. Dhirendra Pratap Singh** and **Dr. Jaytrilok Choudhary**.

We are also grateful to our respected director **Dr. N. S. Raghuwanshi** for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement.

ABSTRACT

Hand recognition is a very active research topic due to its potential applications in the many fields such as automotive interactions, consumer electronics, home automation, conversation with specially abled persons and many more.

Despite the significant improvements, hand recognition is still a challenging problem that waits for more and more accurate algorithms. Classical machine learning approaches often require a complex feature extraction process and produce poor results.

This project presents a model that is capable of recognizing basic hand movement of counting number of fingers raised and performing simple arithmetic operation on them by using deep Convolutional Neural Network (CNN) concepts. The goal of this project is to build an interactive system for little kids, who find it hard to deal with numbers during their elementary education by aiding them to play with numbers on their fingers. The CNN model is generated by using the Python environment and also tried and tested on the same environment.

CONTENTS

Declaration 2

Acknowledgement 3

Abstract 4

List of Figures 6

1. Introduction 7
   - 1.1 History
   - 1.2 Application
   - 1.3 Issues
   - 1.4 Objective

2. Literature reviewandsurvey 10
   - 2.1 SummarizedReview

3. Gaps Identified 12

4. Proposed WorkandMethodology 12
   - 4.1 Overview
   - 4.2 Methodology
   - 4.3 GUI

5. ImplementationandResults 16
   - 5.1 CodeSnippets
   - 5.3 Results and Outputs

6. Conclusion

7. Limitation andFutureScope 28

7. References 29

LIST OF FIGURES

SerialNo.	FigureName	PageNo.
1	DifferentHand Gestures	7
2	CNNArchitecture	8
3	Grahamscan algorithm	11
4	Convexhull	11
5	Flowchart	13
6	Comparison ofdifferent algorithms	27

1. Introduction

Hand recognition is the process of identifying human hand gestures based on hand positions, finger position, angle and shape. In humans, this ability is considered one of the most important social skills. There is some general universality in hand gestures of humans in performing certain actions. Therefore, if properly being modelled, this universality can be a convenient feature in human-machine interaction: a well-trained system can understand handgestures.

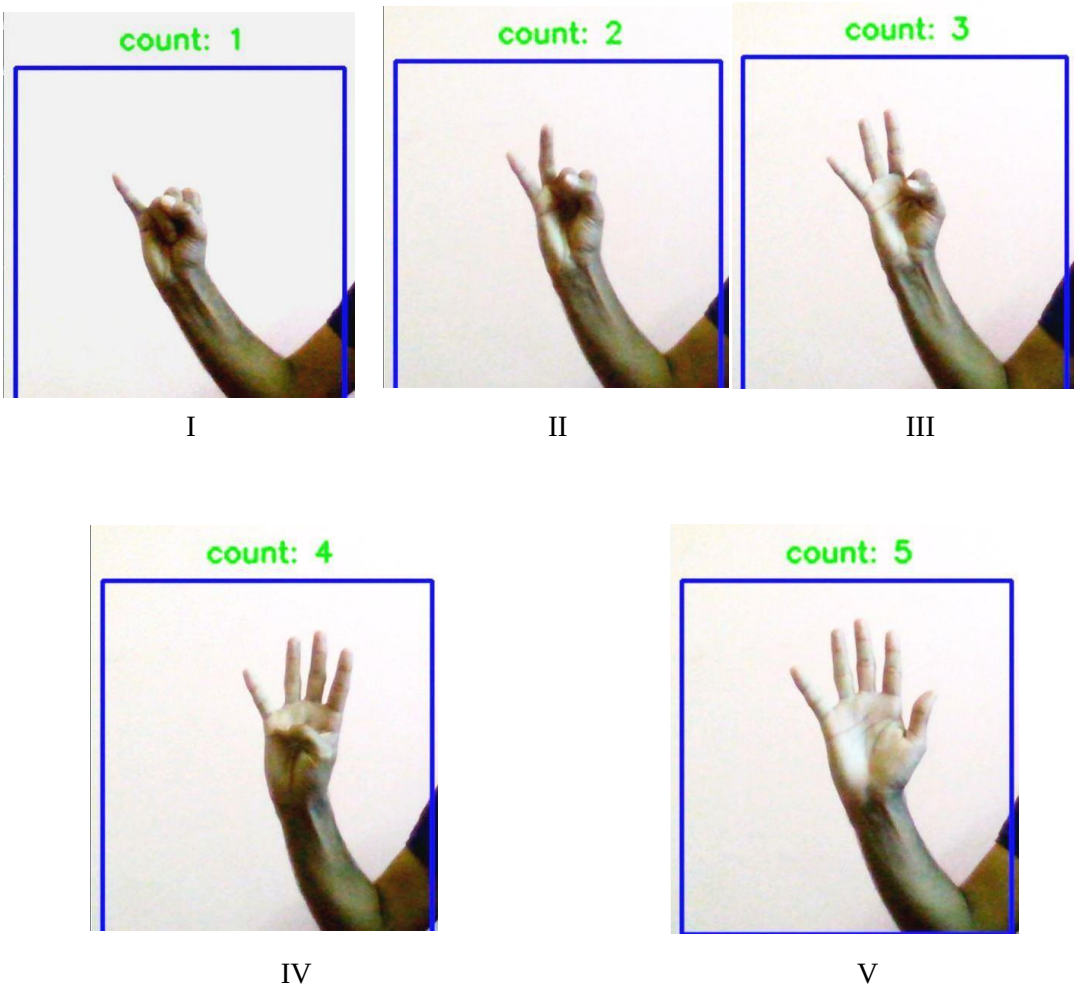

Fig. 1: Different Hand Gestures

1.1 History

Ever since computers were invented, people have wanted to build artificially intelligent (AI) systems that are mentally and/or physically equivalent to humans. In the past decades, the increase of generally available computational

power provided a helping hand for developing fast learning machines, whereas the Internet supplied an enormous amount of data for training. Among a lot of advanced machine learning techniques that have been developed so far, deep learning is widely considered as one of the most promising techniques to make AI machines approaching human-level intelligence and hand movement recognition has attracted considerable attention in the past decade due to their potential applications.

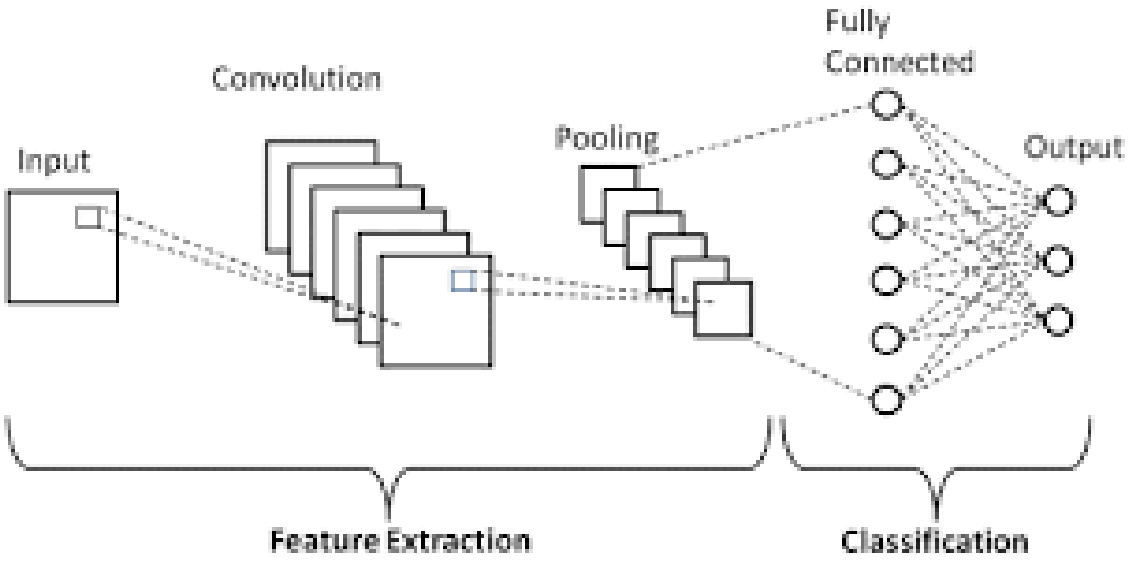

Fig. 2: CNN Architecture

1.2 Application

Automated hand gesture recognition has numerous practical applications such as automotive interactions, consumer electronics, home automation, conversation with specially abled persons and health-care. The ability to read hand gestures and then recognize human actions provides a new dimension to human-machine interactions, for instance, if our system can detect number of fingers raised then for each number detected, we can set a particular instruction for our machine to perform. Robots can also benefit from automated hand gesture recognition. If robots can predict human actions, they can react upon this and have appropriate behaviours. Along with these applications the future goal of this project is to develop a model that can automate gear-change in cars by detecting number of fingers raised through hand gesture detecting techniques.

1.3 Issues

Because of differentiating gestures, hand action recognition is still a challenging task for computers. Gestures from two different hands with the same action may be different, while gestures from one person's hand with two actions may be the same. For example, three fingers can be raised in two ways, either by raising index finger, middle finger and ring finger or by raising index finger, ring finger and pinkie fingers respectively. Information from a hand position is distributed in different areas of hand and depends on orientation, angle and number of fingers raised. Thus, different gesture for same action causes anomality.

1.4 Objective

The main contribution of the proposed is to model hand actions on static images for recognizing five basic hand positions (one finger raised as 1, two fingers raised as 2, three fingers raised as 3, four fingers raised as 4, five fingers raised as 5) and performing basic arithmetic operation on them.

2. Literature

2.1 SummarizedReview

Hand gesture recognition system received great attention in the recent few years because of its manifoldness applications and to interact with machine efficiently through human computer interaction. Most of the researchers classifies gesture recognition system into mainly three steps after acquiring the input image: Extraction method, Feature extraction and Classification.

There are many approaches for gesture recognition which are based on hand-engineered features. A number of methods for gesture recognitions classify the hand image into some basic actions like good, bad, up and down. Various works have been done on hand gesture recognition and some notable research in this topic is mentioned.

A hand gesture recognition system using an artificial neural network (ANN) based on shape fitting technique has been developed. In this system, a colour segmentation technique on YcbCr colour space was used after filtering to detect hand. Then the shape of the hand was approached by the hand morphology. The shape of hands and finger orientation features were extracted and passed to an ANN. They achieved 94.05% accuracy by using this method.

Gesture detection by using an ANN has also been developed. In this system, images were segmented based on skin colours. The selected features for ANN were changes of pixel through cross sections, boundary and the scalar description like aspect ratio and edge ratio. After establishing those feature vectors, they were fed to the ANN for training. The accuracy was around 98%.

Another development was to recognize American Sign Language using neural networks algorithm. The input image are converted into HSV color model, resized into 80x64 and some image preprocessing operations are applied to segment the hand from a uniform background, features are extracted using histogram technique and Hough algorithm. Feed forward Neural Networks with three layers are used for gesture classification. 8 samples are used for each 26 characters in sign language, for each gesture, 5 samples are used for training and 3samples for testing, the system achieved 92.78% recognition rate using MATLABlanguage.

(For Convex Hull)
Graham Scan computes the convex hull of any given set of points in O(nlogn). In graham scan first the Algorithm starts by sorting the set of points by increasing order of x-axis. In its second step it checks the orientation at each

point with two most recently selected points. If the points form left-hand turn (positive orientation) then the points are pushed to the stack but if the point is failed to form a positive orientation with last two pushed points then the last

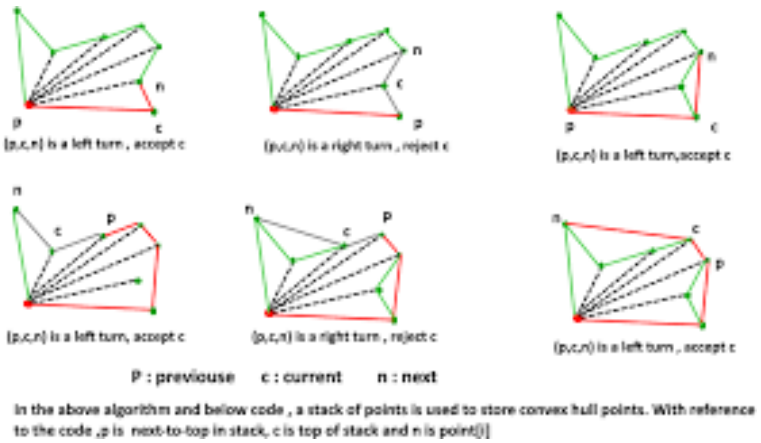

Fig. 3: Graham Scan Algorithm

two points are popped from the stack. Now the join the points stored in the stack, the resultant will be the requiredhull.

The next algorithm presented is the generalization of the sorting procedure Quick sort, named as Quick hull. Quick hull operates in O(nlogn) time but in worst case it can be $O(n^2)$. The basic idea behind quick hull is to discard the points as quickly as possible. In quick hull the first the algorithm calculate the points with minimum and maximum x and y-coordinates and by joining these points a bounding rectangle is defined, within which the hull is contained. Furthermore the points within this quadrilateral are discarded for any further consideration. Now in the continuation of this algorithm the remaining points are classified into four remaining corner triangles. On these corner points now the next task is to find a point among the points on the hull, discard the points within the new triangle and reconsider the two newly formed subsets. This point can be selected by considering possible perpendicular distance from theedge.

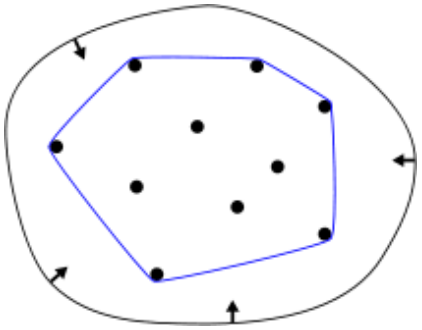

Fig. 4: Convex Hull

3. GapsIdentified

Various proposals discussed above proposed methods on segmentation or feature extraction. Before applying Convolutional Neural Network(CNN), it is common to apply convex hull algorithms to draw the boundary of objects. For finding convex hulls so many algorithms areused.

In this project a hybrid method is proposed to compute convex hull. The method is based on two already existing convex hull algorithms i.e. quick hull and grahams Scan algorithm. The proposed technique is an attempt to remove the deficiencies in the two above mentioned techniques of the convexhull.

4. Proposed work andmethodology

4.1 Overview

The main contribution of the proposed work is to propose a hybrid method to compute convex hull. The method is based on two already existing convex hull algorithms i.e. quick hull and grahams Scan algorithm. The proposed technique is an attempt to remove the deficiencies in the two above mentioned techniques of the convexhull.

The database used is already pre-processed for training of a deep CNN to classify different numbers of fingers raised. Deep learning has appeared as a promising one to perform hand gesture recognition as it extracts both low level and high-level features without any specific method.

4.2 Methodology

The proposed technique is composed of two existing algorithms Quick Hull and Graham scan. To overcome the draw backs of these two algorithms the proposed techniques works. As in Graham Scan , it works quite slowly because it checks the orientation of all the points in the set, for smaller point set the algorithm will definitely perform faster, but for a larger set the computational cost will be really unaffordable. To reduce the set of point's candidate to be on hull the quick hull's initial step is applied as the useless points are discarded at the very initial stage and no extra calculation is performed which will increase the speed of the Graham's method.As discussed earlier Graham Scan selects the points on the basis of their orientation. It checks the orientationoftheselectedpointwithrespecttothelasttwoaddedpointslike

Jarvis method it works quite slowly. The other algorithm (Quick Hull) starts by calculating the points with minimum and the maximum x-and y-coordinates.

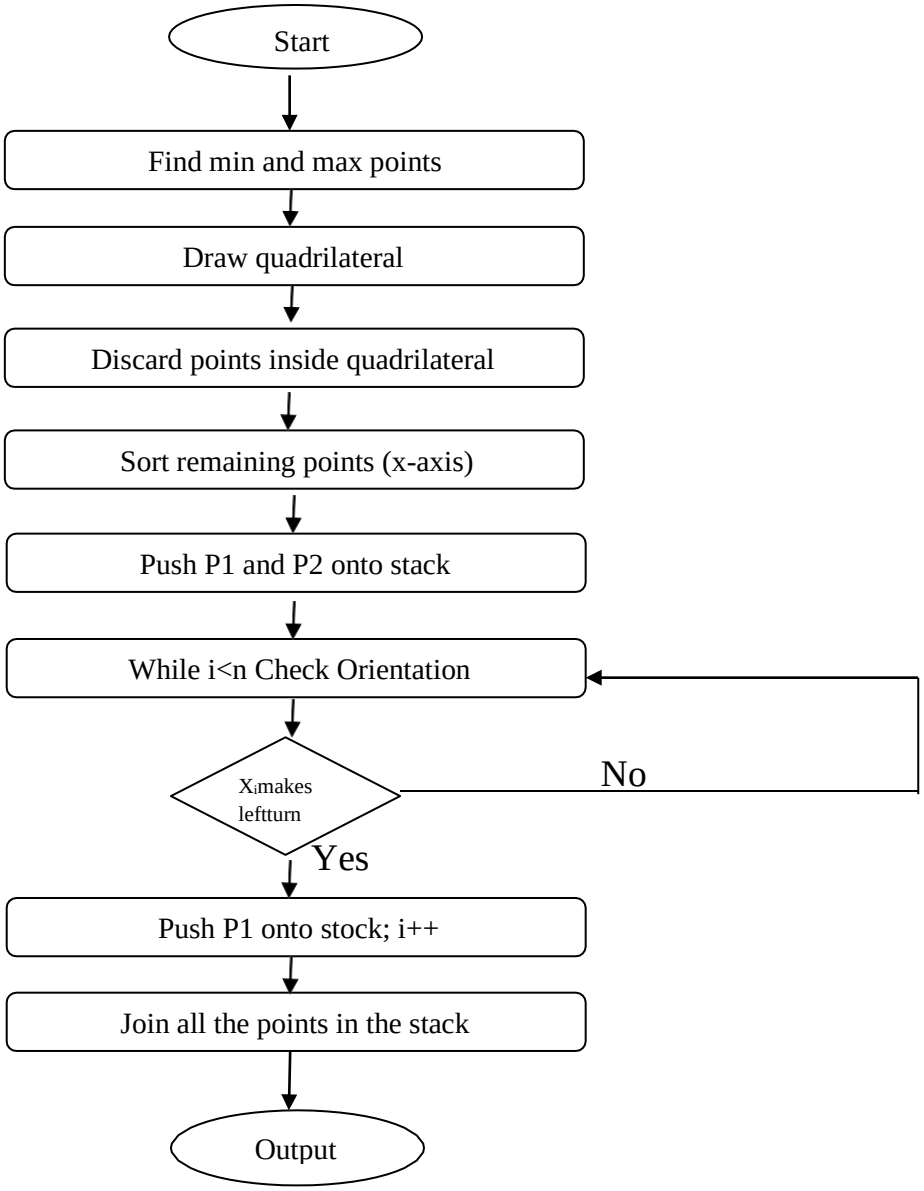

Fig. 5: Flow chart of new algorithm

The proposed technique is explained by using the following flow (see figure 1) chart. The flow chart will describe each step of the algorithm. In the each state of the flowchart each step of the technique is performed. The proposed technique starts with quick hull and in its second step graham scan is applied on remaining points. In the first step of this algorithm, starting with the quick hull, mark the points with minimum and maximum x- and y- coordinates Using these points draw the quadrilateral and discard the points inside this quadrilateral. Now the remaining points are classified into the fourcorner

triangles. Considering the points on one of these corner triangles, sort the remaining points with respect to the x-coordinates and then on these points apply graham scan method to compute the hull. The flow chart has six main states. The first state will find out the minimum and maximum x and y-coordinates as the initial step of the algorithm. In the next state a quadrilateral is drawn, the state next to it will discard the points lying inside thisquadrilateral.

1. Find out the points with Xmin, Xmax, Ymin, and Ymax from setn
2. Construct Quadrilateral using thesepoints
3. Discard the points inside it forming a new $setn_{new}$
4. Sort the points (respect to x-axis), denoted(P_1,P_2,…….P_n)
5. Push $P_1$and P_2 ontoU(stack)
6. Whilei≤n

{
If (xi makes left turn relative to top 2 items on stack)
{
Push xi;
i++;
}
Else
{
Pop (last two added points);
Discard;
}}

The fourth state will sort the points according to the increasing order of their x-coordinates. In the next state from array of points formed in the above state, the two points with lowest x-coordinate (P1, P2) are pushed into the stack and in the sixth state a while loop is used by the help of which the orientation of the points relative to the last two points added to the stack is going to be checked. As it is discussed earlier the orientation can be calculated by taking the cross product of the points. If the points make a left hand turn it means they have a positive orientation and the points are pushed into the stack, and if the points are making a positive orientation or they are linear the point will be discarded and the last two added points in the stack are popped. The technique is quite simple and easy tounderstand.

4.3 GUI

Python provides various options for developing graphical user interfaces (GUIs).

4.3.1 TkinterProgramming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All we need to do is perform the following steps −

- Import the *Tkinter*module.
- Create the GUI application mainwindow.
- Add one or more of the above-mentioned widgets to the GUIapplication.
- Enter the main event loop to take action against each event triggered by theuser.

4.3.2 Tkinter GeometryManagement

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method− This geometry manager organizes widgets in blocks before placing them in the parentwidget.
- The *grid()* Method− This geometry manager organizes widgets in a table-like structure in the parentwidget.
- The *place()* Method− This geometry manager organizes widgets by placing them in a specific position in the parentwidget.

5. Implementation andResults

5.1 Coding Snippets

```
import tkinter
import cv2
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import pairwise

def capture():
    num=0
    cap=cv2.VideoCapture(0)
    i=0
    _,frame=cap.read()
    back=None
    roi=cv2.selectROI(frame)
    (x,y,w,h)=tuple(map(int,roi))
    while True:
        _,frame=cap.read()
        if i<50:
            i+=1
            if back is None:
                back=frame[y:y+h,x:x+w].copy()
                back=np.float32(back)
```

```
        else:
            cv2.accumulateWeighted(frame[y:y+h,x:x+w].copy(),back,0.2)
    else:
            #print(back.shape,frame.shape)
        back=cv2.convertScaleAbs(back)
        back_gray=cv2.cvtColor(back,cv2.COLOR_BGR2GRAY)
        frame_gray=cv2.cvtColor(frame[y:y+h,x:x+w],cv2.COLOR_BGR2GRAY)

        img=cv2.absdiff(back_gray,frame_gray)
        _,img=cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
        con,hie=cv2.findContours(img,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        img2=img.copy()
        con=max(con,key=cv2.contourArea)
        conv_hull=cv2.convexHull(con)
        cv2.drawContours(img,[conv_hull],-1,225,3)

        top=tuple(conv_hull[conv_hull[:,:,1].argmin()][0])
        bottom=tuple(conv_hull[conv_hull[:,:,1].argmax()][0])
        left=tuple(conv_hull[conv_hull[:,:,0].argmin()][0])
        right=tuple(conv_hull[conv_hull[:,:,0].argmax()][0])
        cx=(left[0]+right[0])//2
        cy=(top[1]+bottom[1])//2
```

```
        dist=pairwise.euclidean_distances([left,right,bottom,top],[[cx,cy]])[0]
        radi=int(0.80*dist)

        circular_roi=np.zeros_like(img,dtype='uint8')
        cv2.circle(circular_roi,(cx,cy),radi,255,8)
        wighted=cv2.addWeighted(img.copy(),0.6,circular_roi,0.4,2)

        mask=cv2.bitwise_and(img2,img2,mask=circular_roi)
                #mask
        con,hie=cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NC

        circumfrence=2*np.pi*radi
        count = 0
        for cnt in con:
            (m_x,m_y,m_w,m_h)=cv2.boundingRect(cnt)
            out_wrist_range=(cy+(cy*0.25))>(m_y+m_h)
            limit_pts=(circumfrence*0.25)>cnt.shape[0]
            if limit_pts and out_wrist_range:
                        #print(limit_pts,out_wrist_range)
                count+=1
```

```
        cv2.putText(frame,'count: '+str(count),(120,70),
                    cv2.FONT_HERSHEY_SIMPLEX ,1,(0,250,0),thickness=4)
        cv2.rectangle(frame,(x,y),(x+w,y+h),255,3)
        cv2.imshow('mask',mask)
        cv2.imshow('frame',frame)
        cv2.imshow('weight',wighted)


    k=cv2.waitKey(30) & 0xff
    if k==27:
        num = count
        break

cap.release()
cv2.destroyAllWindows()
return num
```

```
import one

import tkinter
import cv2
import time
import numpy as np
from tkinter import *
import tkinter.messagebox
import random
import matplotlib.pyplot as plt
from sklearn.metrics import pairwise

##
#for tkiter frame

rt = Tk()
rt.config(background="orange")
rt.geometry("400x780+200+2")
rt.title("calc")

## creating frame ###
top = Frame(rt,width =120,height=10,bg="orange" ,relief = SUNKEN)
top.pack(side=TOP)
left=Frame(rt,width=10,height=50,bg="orange",relief=SUNKEN)
left.pack(side=LEFT)
```

```
right = Frame(rt,width=1230,height=630,bg="green",relief=SUNKEN)
right.pack(side=RIGHT)


##creating gui normal for normal calculator


##

operator = ""
operator1 = ""
d=""# global variable operator
##creating window and its geometry ###

## creating calc ##

txt = StringVar()
txt2 = StringVar()
e1 = Entry(right,textvariable=txt2,width=9,bg="aqua",font=("arial",50),justify='right')
e1.grid(row=2,columnspan=297,rowspan=1,column=1)
e2 = Entry(right,textvariable=txt,width=9,bg="aqua",font=("arial",50),justify='right')
e2.grid(row=1,columnspan=297,rowspan=1,column=1)
```

```
Ll1 = StringVar()
Ll2 = StringVar()
dd = Entry(left,textvariable=Ll1,width=5,bg="aqua",font=("arial",55),justify='left')
dd.grid(row=5,columnspan=295,rowspan=1,column=0)




def l(txt2):
    b(txt2)


def b3():

    global operator1
    global operator
    global d


    if(d=="+"):
        operator=float(operator)
        operator1=float(operator1)
        txt2.set("%d+%d="%(operator1,operator))
        operator=operator+operator1
        txt.set(str(operator))
```

```
def b1(sign):
      global operator1
      global operator
      global d
      operator1=str(operator)
      operator=""
      d=sign
      txt2.set(operator1+sign)

      #txt.set(operator)

def bv():
      if(d!=""):
            b3()


def b(num2):
      global operator
      global operator1
      global d
      operator=str(operator)+str(num2)
      txt2.set(operator)
      txt2.set(operator1+d+operator)
      bv()
#  txt.set(operator)
```

```
def b2():
        global operator
        global operator1

        operator=str(operator)
        operator1=str(operator1)
        operator=(operator[:-1])
        operator1=(operator1[:-1])
        txt.set(str(operator))
        txt2.set(str(operator1))

def b101(num3):
        global operator
        global operator1
        global d
        operator=str(num3)
        operator1=str(num3)
        d=num3
        txt.set(operator)
        txt2.set(operator)

def b102():
        global operator
        global operator1
        L11.set(operator)
```

```
def captureButton():
        num = one.capture()
        b(num)




#============first row button=====#
b11 = Button(right,text="C",padx=14,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
             command =lambda: b101(""))
b11.grid(row=3,column=7)
b12 = Button(right,text="/",padx=17,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
             command=lambda:b1("/"))
b12.grid(row=3,column=8)
b13 = Button(right,text="X",padx=14,pady=7,bd=3,fg="black",bg="yellow",font=("arial",30),
             command=lambda:b1("*"))
b13.grid(row=3,column=9)
b14 = Button(right,text="⌫",padx=1,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
             command=lambda:b2())
b14.grid(row=3,column=10)

#============second row button=====#
b100 = Button(right,text="7",padx=18,pady=9,bd=3,fg= "black",bg="yellow",font=("arial",34),
              command=lambda: b(7))
b100.grid(row=4,column=7)
```

```
b6=Button(right,text="6",padx=12,pady=9,bd=3,fg="black",bg="yellow",font=("arial",34),
          command=lambda: b(6))
b6.grid(row=5,column=9)
b7=Button(right,text="5",padx=12,pady=9,bd=3,fg="black",bg="yellow",font=("arial",34),
          command=lambda: b(5))
b7.grid(row=5,column=8)
b8=Button(right,text="4",padx=18,pady=9,bd=3,fg="black",bg="yellow",font=("arial",34),
          command=lambda: b(4))
b8.grid(row=5,column=7)

#=============fourth row button=====#
b4 = Button(right,text="3",padx=12,pady=1,bd=3,fg="black",bg="yellow",font=("arial",34),
            command=lambda: b(3))
b4.grid(row=6,column=8)
b5 = Button(right,text="2",padx=12,pady=1,bd=3,fg="black",bg="yellow",font=("arial",34),
            command=lambda: b(2))
b5.grid(row=6,column=9)
b5 = Button(right,text="1",padx=18,pady=1,bd=3,fg="black",bg="yellow",font=("arial",34),
            command=lambda: b(1))
b5.grid(row=6,column=7)
b15 = Button(right,text="=",padx=15,pady=48,bd=5,fg= "black",bg="yellow",font=("arial",34),
             command=lambda: b102())
b15.grid(column=10,rowspan=2,row=6)
```

```
b6 = Button(right,text="%",padx=12,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
            command=lambda: b1("%"))
b6.grid(row=7,column=7)
b9= Button(right,text="0",padx=12,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
           command=lambda: b(0))
b9.grid(row=7,column=8)
b10 = Button(right,text=".",padx=18,pady=3,bd=3,fg="black",bg="yellow",font=("arial",34),
             command=lambda: b("."))
b10.grid(row=7,column=9)

#==================top====#
w=Label(left,text="  WELCOME TO OUR PROJECT  ",fg="red",font=("Colonna MT",36,'bold'))
w.grid(row=0,column=0)
local_time=time.asctime(time.localtime(time.time()))
w1=Label(left,font=('DigifaceWide',15,'bold','italic','underline'),text=local_time,
         bg="yellow",width=40,fg="black",bd=10)
w1.grid(row=1,column=0)
ww=Label(left,text="Calculate Easily",fg="orange",bg="orange" ,font=("Colonna MT",36,'bold'))
ww.grid(row=2,column=0)
cc = Button(left,text="open_camera",padx=13,pady=7,bd=3,fg="black",bg="red",font=("arial",36),
            command=lambda: captureButton())
cc.grid(row=4,column=0)
# captureButton = Button(text="Capture", height=6, width=6, bg="black", fg="white", command=cap
# captureButton.place(x=0, y=2)
```

```
import matplotlib.pyplot as plt
import math
import time
import numpy as np
import sys

start_t = time.clock()

def directionOfPoint(A, B, P):


    # Subtracting co-ordinates of
    # point A from B and P, to
    # make A as origin
    B[0] -= A[0]
    B[1] -= A[1]
    P[0] -= A[0]
    P[1] -= A[1]

    # Determining cross Product
    cross_product = B[0] * P[1] - B[1] * P[0]

    # Return RIGHT if cross product is positive
    if (cross_product > 0):
        return 1

    # Return LEFT if cross product is negative
    else:
        return 0
```

```
# points the index of the smallest ordinate point, if there are more than one, return the smallest abscissa point
def get_bottom_point(points):
    min_index = 0
    n = len(points)
    for i in range(0, n):
        if points[i][1] < points[min_index][1] or (
                points[i][1] == points[min_index][1] and points[i][0] < points[min_index][0]):
            min_index = i
    return min_index
```

```
# Sort according to the polar angle with the center point, cosine, center_point: center point
def sort_polar_angle_cos(points, center_point):
    n = len(points)
    cos_value = []
    rank = []
    norm_list = []
    for i in range(0, n):
        point_ = points[i]
        point = [point_[0] - center_point[0], point_[1] - center_point[1]]
        rank.append(i)
        norm_value = math.sqrt(point[0] * point[0] + point[1] * point[1])
        norm_list.append(norm_value)
        if norm_value == 0:
            cos_value.append(1)
        else:
            cos_value.append(point[0] / norm_value)

    for i in range(0, n - 1):
        index = i + 1
        while index > 0:
            if cos_value[index] > cos_value[index - 1] or (
                    cos_value[index] == cos_value[index - 1]
                    and norm_list[index] > norm_list[index - 1]):
                temp = cos_value[index]
                temp_rank = rank[index]
                temp_norm = norm_list[index]
                cos_value[index] = cos_value[index - 1]
                rank[index] = rank[index - 1]
                norm_list[index] = norm_list[index - 1]
                cos_value[index - 1] = temp
                rank[index - 1] = temp_rank
                norm_list[index - 1] = temp_norm
                index = index - 1
            else:
                break
    sorted_points = []
    for i in rank:
        sorted_points.append(points[i])

    return sorted_points
```

```
# Return the angle between the vector and the vector [1, 0]-how many degrees
# to rotate counterclockwise from [1, 0] to reach this vector
def vector_angle(vector):
    norm_ = math.sqrt(vector[0] * vector[0] + vector[1] * vector[1])
    if norm_ == 0:
        return 0

    angle = math.acos(vector[0] / norm_)
    if vector[1] >= 0:
        return angle
    else:
        return 2 * math.pi - angle


# Cross product of two vectors
def coss_multi(v1, v2):
    return v1[0] * v2[1] - v1[1] * v2[0]


def graham_scan(points):
    bottom_index = get_bottom_point(points)
    bottom_point = points.pop(bottom_index)
    sorted_points = sort_polar_angle_cos(points, bottom_point)

    m = len(sorted_points)
    if m < 2:
        print("The number of points is too small to form a convex hull")
        return

    stack = []
    stack.append(bottom_point)
    stack.append(sorted_points[0])
    stack.append(sorted_points[1])
    # print('current stack', stack)
```

```
    for i in range(2, m):
        length = len(stack)
        top = stack[length - 1]
        next_top = stack[length - 2]
        v1 = [sorted_points[i][0] - next_top[0], sorted_points[i][1] - next_top[1]]
        v2 = [top[0] - next_top[0], top[1] - next_top[1]]

        while coss_multi(v1, v2) >= 0:
            if length < 3:     # After adding these two lines of code, no error will be reported when the amount of data is large
                break          # After adding these two lines of code, no errors will be reported when the amount of data is large
            stack.pop()
            length = len(stack)
            top = stack[length - 1]
            next_top = stack[length - 2]
            v1 = [sorted_points[i][0] - next_top[0], sorted_points[i][1] - next_top[1]]
            v2 = [top[0] - next_top[0], top[1] - next_top[1]]
        stack.append(sorted_points[i])

    return stack


# Generate random points
n_iter = [5000, 10000, 15000, 20000, 25000]
time_cost = []
for n in n_iter:
    points = []
    for i in range(n):
        point_x = np.random.randint(1, 100)
        point_y = np.random.randint(1, 100)
        temp = np.hstack((point_x, point_y))
        point = temp.tolist()
        points.append(point)

    #### IMPROVISATION ####
    left = points[0]
    right = points[0]
    top = points[0]
    bottom = points[0]


    for j in range(0, n):
        if(points[j][0] < left[0]):
            left = points[j]
        if(points[j][0] > right[0]):
            right = points[j]
        if(points[j][1] > top[1]):
            top = points[j]
        if(points[j][1] < bottom[1]):
            bottom = points[j]

    newpoints = []
```

```
    for j in range(0, n):
        if(directionOfPoint(top, left, points[i]) == 0):
            newpoints.append(points[i])

        elif(directionOfPoint(right, top, points[i]) == 0):
            newpoints.append(points[i])

        elif(directionOfPoint(left, bottom, points[i]) == 0):
            newpoints.append(points[i])

        elif(directionOfPoint(bottom, right, points[i]) == 0):
            newpoints.append(points[i])



    result = graham_scan(newpoints)

    # Record program running time
    end_t = time.clock()
    time_iter = end_t - start_t
    #print("Graham-Scan algorithm running time:", time_iter)
    # draw(list_points, border_line)
    time_cost.append(time_iter)

# Running time under different test sets
plt.plot(n_iter, time_cost)
plt.show()
```

5.2 Results and Output

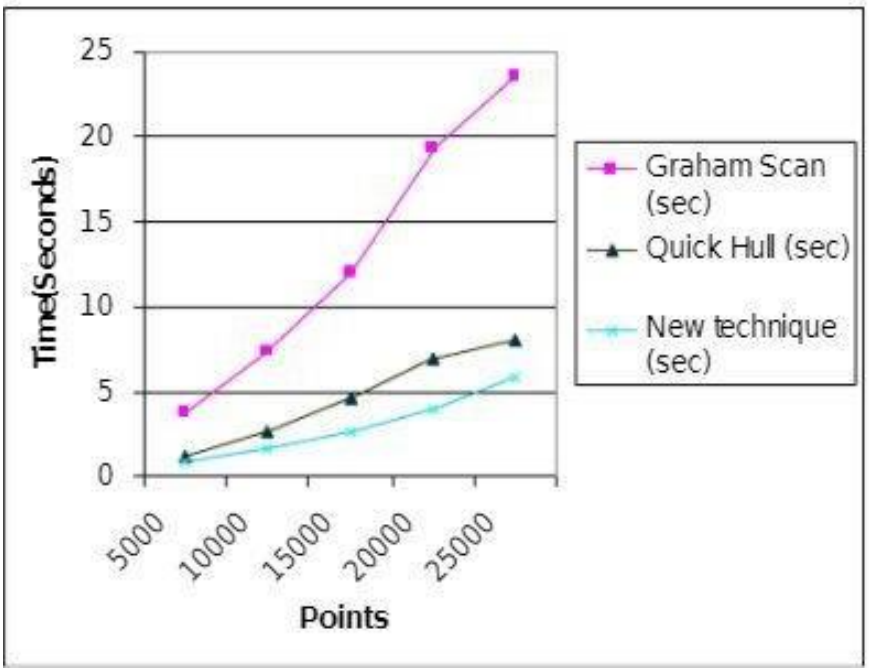

Fig. 6: Comparison of different algorithms

6. Conclusion

Convex hull is a very important term in the field of computational geometry. It is helpful where we need to model the objects with some non deterministic shapes. It has its applications in pattern recognition, image processing and GIS. In this paper a new technique to construct convex hull is presented. The technique can also be referred as the improvement of the graham scan algorithm. The graham scan algorithm performs faster for smaller number of input points. To reduce the input point set the quick hull's initial step is applied tothe pointsdue to which unnecessary calculationsareeliminated and ultimately the task is performed faster and we can have the results faster than before. This is also proved by the results that the proposed technique is faster than any of two algorithms. As discussed earlier the main problem with the graham scan is that it has to make calculations on every single point in the set. Main ideabehindthis approach is to reduce the number of calculations, as more points are discarded in the initial step. Using quick hull, the points in the input set can bereduced.

7. References

1. JonathanFidelisPaul,DibyabivaSeth,CijoPaul,andJayatiGhosh Dastidar, “Hand Gesture Recognition Library”, International Journal ofScience and Applied Information Technology, Volume 3
2. SwapnilD.Badgujar,GourabTalukdar,OmkarGondhalekarandMrs. S.Y. Kulkarni, “Hand Gesture Recognition System”, International Journal of Scientific and Research Publications, Volume4
3. Abhishek, S, K., Qubeley, Fai, L. C., Ho, & Derek. (2016). Glove-based hand gesture recognition sign language translator using capacitive touch sensor. In 2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC) (pp. 334- 337):IEEE
4. Dipietro, L., Sabatini, A. M., & Dario, P. (2008). A Survey of Glove-Based Systems and Their Applications. Ieee transactions on systems, man, andcybernetics
5. https://www.geeksforgeeks.org/quickhull-algorithm-convex-hull/
6. https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/
7. https://opencv.org/