*HAND GESTURE RECOGNITION*

*TO COMMAND A COMPUTER*

Submitted by: Rohit Gupta

Roll. No: 160BTCCSE013

**Introduction**

The basic idea behind this project is to recognize the hand gestures of a human. Each gesture represents a command that can either be defined in a separate python script or in an Arduino program.

This project makes use of OpenCV to identify hand gestures. Firstly, a Region-of-Interest (ROI) is defined. Then, as soon as, a hand is displayed in front of ROI, hand contour is found. This project counts the number of fingers, then this count is displayed using Arduino by glowing the same number of LEDs as the number of fingers detected. The algorithm finds the largest possible circular palm contour and the number of vertical intersections at the circumference of the circular contour indicates number of fingers detected.

These numbers can be programmed and made available for many uses. The numbers can be used to command a Robot, or to control a computer, or to simulate a video or even in a door locking / unlocking and many more uses possible…

**What is needed?**

Arduino

OpenCV Knowledge

Python

Spyder / Jupyter (Anaconda distribution)

Few LEDs

Resistors (optional)

**Code – recognize.py**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Thu Nov 8 20:27:48 2018

@author: rohitgupta

"""

import cv2

import imutils

import numpy as np

from sklearn.metrics import pairwise

import serial

import time

arduino = serial.Serial("/dev/cu.usbmodem1411", 9600)

time.sleep(2)

bg = None

#Function to find the running average over the background

def run\_avg(image, aWeight):

global bg

if bg is None:

bg = image.copy().astype("float")

return

cv2.accumulateWeighted(image, bg, aWeight)

#Function to segment the region of hand in the image

def segment(image, threshold=25):

global bg

diff = cv2.absdiff(bg.astype("uint8"), image)

thresholded = cv2.threshold(diff,

threshold,

255,

cv2.THRESH\_BINARY)[1]

(\_,cnts,\_) = cv2.findContours(thresholded.copy(),

cv2.RETR\_EXTERNAL,

cv2.CHAIN\_APPROX\_SIMPLE)

if len(cnts) == 0:

return

else:

#based on contour area, get the maximum contour which is the hand

segmented = max(cnts, key=cv2.contourArea)

return (thresholded, segmented)

#Function to count the number of fingers in the segmented hand region

def count(thresholded, segmented):

chull = cv2.convexHull(segmented)

extreme\_top = tuple(chull[chull[:,:, 1].argmin()][0])

extreme\_bottom = tuple(chull[chull[:,:, 1].argmax()][0])

extreme\_left = tuple(chull[chull[:,:, 0].argmin()][0])

extreme\_right = tuple(chull[chull[:,:, 0].argmax()][0])

#find the center of the palm

cX = int((extreme\_left[0] + extreme\_right[0])/2)

cY = int((extreme\_top[1] + extreme\_bottom[1])/2)

distance = pairwise.euclidean\_distances([(cX, cY)],

Y=[extreme\_left, extreme\_right, extreme\_top, extreme\_bottom])[0]

maximum\_distance = distance[distance.argmax()]

radius = int(0.8 \* maximum\_distance)

#find the circumference of the circle

circumference = 2 \* np.pi \* radius

#take out the circular region of interest which has the palm and the fingers

circular\_roi = np.zeros(thresholded.shape[:2], dtype="uint8")

cv2.circle(circular\_roi, (cX, cY), radius, 255, 1)

circular\_roi = cv2.bitwise\_and(thresholded, thresholded, mask=circular\_roi)

(\_, cnts, \_) = cv2.findContours(circular\_roi.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

count = 0

for c in cnts:

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

if((cY + (cY \* 0.25)) > (y+h)) and ((circumference \* 0.25) > c.shape[0]):

count += 1

return count

#Main function

if \_\_name\_\_ == "\_\_main\_\_":

aWeight = 0.5

camera = cv2.VideoCapture(-1)

top, right, bottom, left = 10, 350, 225, 590

num\_frames = 0

calibrated = False

while(True):

#get the current frame

(grabbed, frame) = camera.read()

frame = imutils.resize(frame, width=700)

frame = cv2.flip(frame, 1)

#clone the frame

clone = frame.copy()

(height, width) = frame.shape[:2]

roi = frame[top:bottom, right:left]

gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (7,7), 0)

if num\_frames < 30:

run\_avg(gray, aWeight)

if num\_frames == 1:

print("[STATUS]Please wait, calibrating...!!!")

elif num\_frames == 29:

print("[STATUS]Calibration Successfull!")

else:

hand = segment(gray)

if hand is not None:

(thresholded, segmented) = hand

#draw the segmented region and display the frame

cv2.drawContours(clone, [segmented + (right, top)], -1, (0,0,255))

fingers = count(thresholded, segmented)

arduino.write((str(fingers)).encode())

time.sleep(1)

print(fingers)

cv2.putText(clone, str(fingers), (70, 45), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0,0,255), 2)

cv2.imshow("Thresholded", thresholded)

cv2.rectangle(clone, (left, top), (right, bottom), (0, 255, 0), 2)

num\_frames += 1

cv2.imshow("Video feed", clone)

keypress = cv2.waitKey(1) & 0xFF

if keypress == ord("q"):

break

camera.release()

cv2.distroyAllWindows()

**Code – recognize.ino**

char data;

void setup()

{

Serial.begin(9600);

pinMode(LED\_BUILTIN, OUTPUT);

pinMode(7, OUTPUT);

pinMode(6, OUTPUT);

pinMode(5, OUTPUT);

digitalWrite (LED\_BUILTIN, LOW);

digitalWrite(7, LOW);

digitalWrite(6, LOW);

digitalWrite(5, LOW);

}

void loop()

{

while(!Serial.available()) {}

while (Serial.available())

{

data = Serial.read();

Serial.println(data);

}

if(data == '1') {

digitalWrite(7, HIGH);

}

else if(data == '2') {

digitalWrite(7, HIGH);

digitalWrite(6, HIGH);

}

else if(data == '3') {

digitalWrite(7, HIGH);

digitalWrite(6, HIGH);

digitalWrite(5, HIGH);

}

else {

// digitalWrite(7, HIGH);

// digitalWrite(6, HIGH);

// digitalWrite(5, HIGH);

digitalWrite(LED\_BUILTIN, HIGH);

}

digitalWrite(7, LOW);

digitalWrite(6, LOW);

digitalWrite(5, LOW);

digitalWrite(LED\_BUILTIN, LOW);

}

\*\*\*