VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRICAL ENGINEERING

Non-Contact Number Detection for ATMs And Elevators to Prevent Chances of Covid-19 Transmission

Done by

SRAWAN PRADHAN
(17BEI0058)

SHAHNAWAZ AHMAD
(17BEI0017)

SHAURYA DAS

(17BEI0032)

RAHUL KUMAR

(17BEI0085)

Slot: A1

For the course Course Code: EEE1008

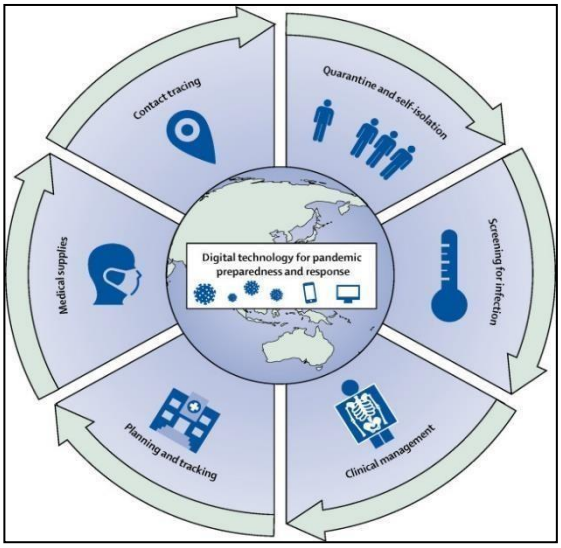
Course Name: BIOMEDICAL INSTRUMENTATION

Semester: Fall 2020-21

Content

ABSTRACT

The unparalleled outbreak of 2019 novel corona virus termed as COVID 19 by the World Health organization (WHO) has placed the governments of different countries in an unsecure position. The number of confirmed corona cases has been increasing at an alarming rate throughout the world. In response to such acts, numerous attempts to reduce the COVID transmission has been made in the form of social distancing, wearing masks and use of sanitizers. Other people can catch COVID-19 when the virus gets into their mouth, nose or eyes, which is more likely to happen when people are in direct or close contact (less than 1 meter apart) with an infected person. Transmission can take place when people come in contact with public surfaces like ATM machine buttons, door bells etc.



Current evidence suggests that the main way the virus spreads is by respiratory droplets among people who are in close contact with each other. Aerosol transmission can occur in specific settings, particularly in indoor, crowded and inadequately ventilated spaces, where infected person(s) spend long periods of time with others, such as restaurants, choir practices, fitness classes, nightclubs, offices and/or places of worship. More studies are underway to better understand the conditions in which aerosol transmission is occurring outside of medical facilities where specific medical procedures, called aerosol generating procedures, are conducted. We aim at using image processing to automatically detect numbers from fingertips to prevent chances of COVID transmission.

With the help of Image Processing, we strive to implement a system that will be able to detect fingertips so that a person need not touch a surface to interact with the system. The proposed system will be able to prevent the spread of COVID as it directly eliminates the need to be in contact with surfaces that have been used by multiple people. The said system will have a wide range of applications from ATMs to elevators – all of these places that have been used often by the public and could potentially be a place prone area.

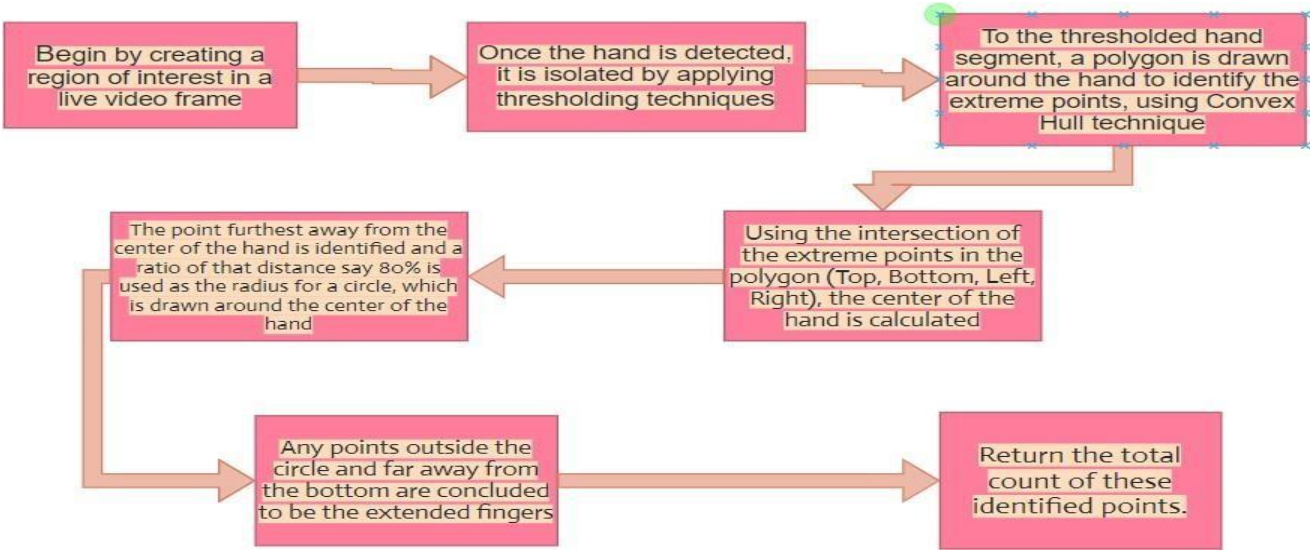
INTRODUCTION

This project aims to reduce the chances of contracting viruses and germs by means of touch. With the help of image processing, one can easily accomplish contact less machine-human interaction. As computers get faster, implementing this becomes easier and more efficient. The only requirements would be a functional camera and a computer.

The proposed system starts by defining a region of interest. This is the area in which all capture will take place. Thresholding techniques will be used to find the contours of the user's hand and fingers. Using the convex hull technique, a polygon is drawn around the hand which helps identify the extreme points. Once the extreme points have been calculated using this technique, the intersection of these points is taken as the center of the user's hand. Further processing is carried out to define the contour of the hand, which in turn defines the fingers.

It is aimed to find out the number of fingers drawn accurately as this is the basis of the proposed project. Performing this task in a matter of few seconds is vital as number specific application such as ATMs demand it. It is also a requirement to have error free outputs as the working of these machines depend on it.

BLOCK DIAGRAM AND EXPLANATION



1. Begin by obtaining a region of interest in a video live frame, where the object (hand in our case) is to be inserted for counting.

2. Once the hand (object) is detected, it is isolated by applying thresholding techniques, Binary Thresholding in this case using OpenCV.

3. To the thresholder hand segment, a polygon is drawn around the hand to identify the extreme points, using Convex Hull technique (Gift wrapping algorithm).

4. Using the intersection of the extreme points in the polygon (Top, down, Right and left), the centre of the hand is calculated.

5. The point furthest away from the centre of the hand is identified and a ratio of that distance say 80% is used as the radius for a circle, which is drawn around the centre of the hand (For visualization purposes, we may think of this as the palm region).

6. Any points outside the circle and far away from the bottom are considered to be the extended fingers.

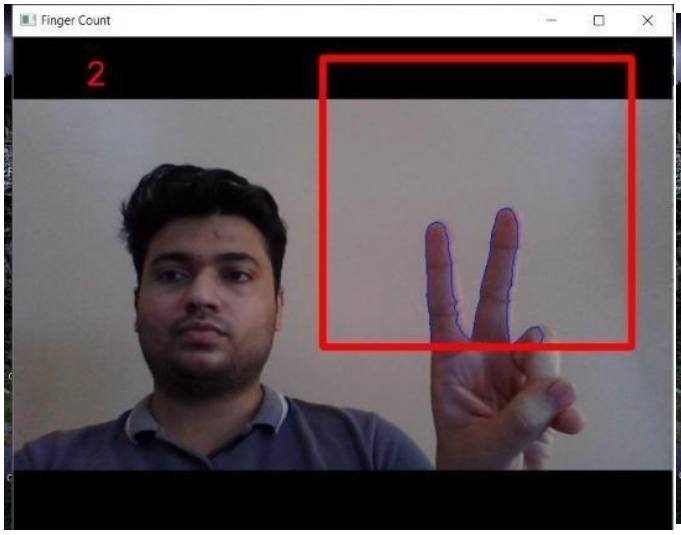
7. Return the total count of these of the points (in our case fingers).

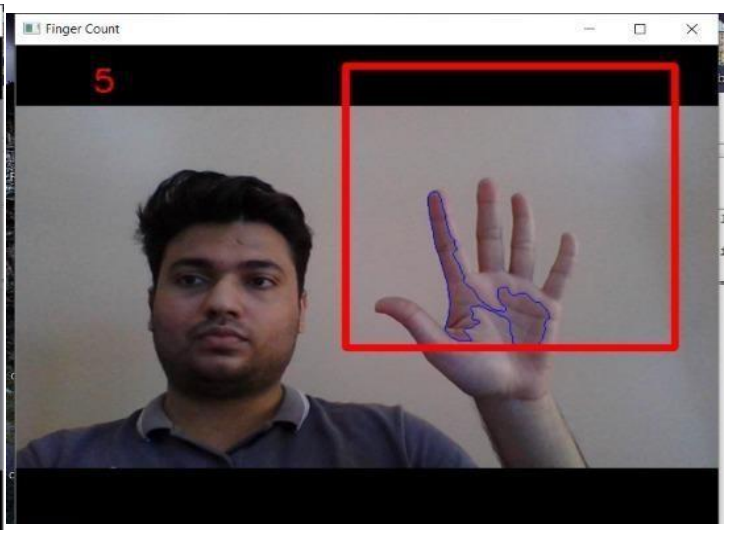
SOFTWARE DESIGN

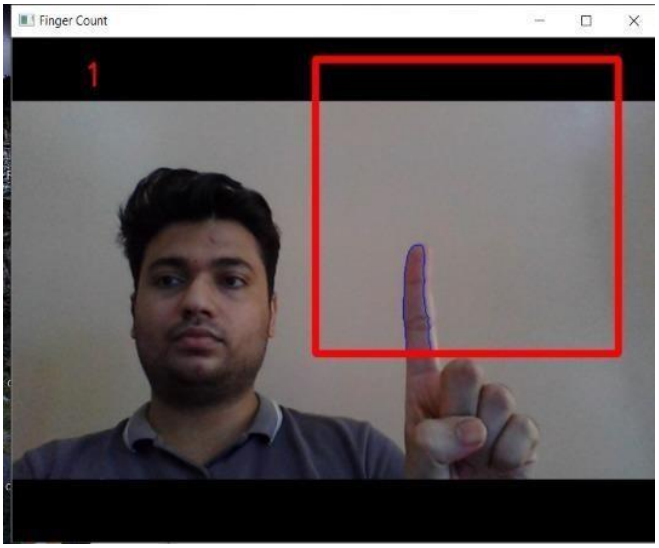
```
import cv2
import numpy as np
from sklearn.metrics import pairwise
background = None
accumulated_weight = 0.5
roi_top = 20
roi_bottom = 300
roi_right = 300
roi_left = 600
def calc_accum_avg(frame, accumulated_weight):
if background is None:
        background = frame.copy().astype("float")
        return None
    cv2.accumulateWeighted(frame, background, accumulated_weight)
def segment(frame, threshold=25):
    global background
    diff = cv2.absdiff(background.astype("uint8"), frame)
   thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)
    image, contours, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    if len(contours) == 0:
        return None
    else:
        hand_segment = max(contours, key=cv2.contourArea)
        return (thresholded, hand_segment)

def count_fingers(thresholded, hand_segment):
    conv_hull = cv2.convexHull(hand_segment)
    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
    distance = pairwise.euclidean_distances([(cX, cY)], Y=[left, right, top, bottom])[0]
    max_distance = distance.max()
    radius = int(0.8 * max_distance)
    circumference = (2 * np.pi * radius)
    circular_roi = np.zeros(thresholded.shape[:2], dtype="uint8")
    cv2.circle(circular_roi, (cX, cY), radius, 255, 10)
    circular_roi = cv2.bitwise_and(thresholded, thresholded, mask=circular_roi)
    image, contours, hierarchy = cv2.findContours(circular_roi.copy(), cv2.RETR_EXTERNAL,
```

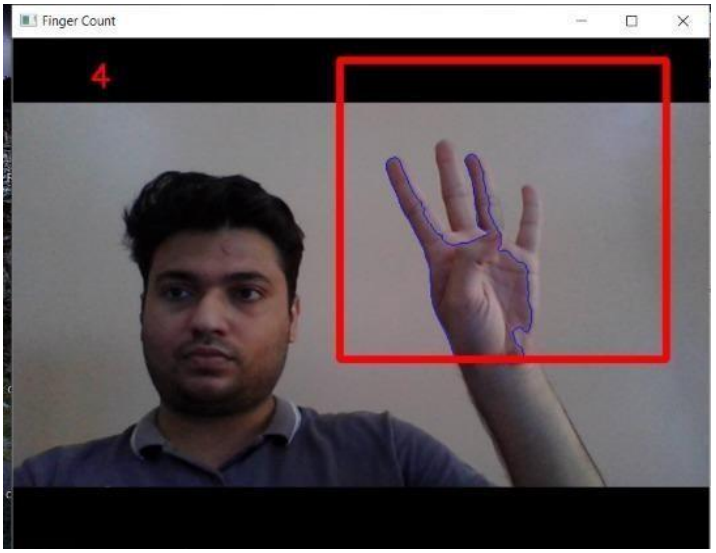
```
cv2.CHAIN_APPROX_NONE)
    count = 0
    for cnt in contours:
        (x, y, w, h) = cv2.boundingRect(cnt)
        out_of_wrist = ((cY + (cY * 0.25)) > (y + h))
        limit_points = ((circumference * 0.25) > cnt.shape[0])
        if out_of_wrist and limit_points:
            count += 1
    return count
cam = cv2.VideoCapture(0)
num_frames = 0
while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame, 1)
    frame_copy = frame.copy()
    roi = frame[roi_top:roi_bottom, roi_right:roi_left]
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (7, 7), 0)
    if num_frames < 60:
        calc_accum_avg(gray, accumulated_weight)
        if num_frames <= 59:
            cv2.putText(frame_copy, "WAIT! GETTING BACKGROUND AVG.", (200, 400),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            cv2.imshow("Finger Count",frame_copy)
    else:
        hand = segment(gray)
        if hand is not None:
            thresholded, hand_segment = hand
            cv2.drawContours(frame_copy, [hand_segment + (roi_right, roi_top)], -1, (255, 0, 0),1)
            fingers = count_fingers(thresholded, hand_segment)
            cv2.putText(frame_copy, str(fingers), (70, 45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255),
2)
            cv2.imshow("Thesholded", thresholded)
    cv2.rectangle(frame_copy, (roi_left, roi_top), (roi_right, roi_bottom), (0,0,255), 5)

    num_frames += 1
    cv2.imshow("Finger Count", frame_copy)
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break
cam.release()
cv2.destroyAllWindows()
```

SCREENSHOTS









CONCLUSION

In this project we have developed the code in which we count the number of fingers. we implemented this along with the camera that is already present in the system and python. We used OpenCV and Convex hull methods as our main tools in this project. This project works fine for any kind of hands skin color, as it was need in country like India which is having heterogenous population. This works in low light conditions also; the only concern is that the background calibration and stabilization must be done with maximum precession and care. This project has various applications. They can be applied to vending machines as well. If we use this in atm so we need to have a cover around hand so our Pin is not leaked out. This project implemented can save the spread of COVID-19 from Elevators, Atm and etc. These are some area we touch every day and it is publicly used. Care at these places is must to have as one infected can affect the whole society.

With the low processing power and fast results this project has a high scope of applications at places that involve a high number of people. Once the pandemic ends, this project can be further used in suggested areas to maintain hygiene. This would further prevent the spread of other germs and viruses that spread via contact.

Link:
https://drive.google.com/file/d/1s-LOSTDc1Z7luub1hdRZHFNLAta4KS5d/view?usp=sharing

FUTURE SCOPE

This is our little contribution in fighting the COVID-19 in these unprecedented times one can utilize technology in fighting the virus. There can lot to done using image detection which can make things more contactless.

1. We can use OpenCV to detect mask and connect them with doors which only opens if a person is wearing a mask or not.
2. We can also detect gestures instead of fingers instead of public touchscreen installed at airports and railways. Which can make our boarding more hassle free.
3. We can also use OpenCV and this convex Hall technique to detect distance between two peoples, if they are maintaining social Distancing or not.
4. Additional security features can be added to ensure no data leaks occur.
5. A standalone version can be implemented for modularity.

REFERENCES

1. Nguyen, Linh and Song, Chanyoung and Ryu, Joonghyun and An, P. and Hoang, Nam-Dũng and Kim, Deok-Soo. (2019). QuickhullDisk: A faster convex hull algorithm for disks. Applied Mathematics and Computation. 363. 124626. 10.1016/j.amc.2019.12462
2. Mei, G. CudaChain: an alternative algorithm for finding 2D convex hulls on the GPU. SpringerPlus 5, 696 (2016)
3. Jayaramaiah Boreddy, An incremental computation of convex hull of planar line intersections, Pattern Recognition Letters,Volume 11, Issue 8,1990, Pages 541-543, ISSN 0167-8655
4. Gaj Vidmar, Maja Pohar,Borgwardt, K.H. Average complexity of a gift-wrapping algorithm for determining the convex hull of randomly given points. Discrete Comput Geom 17, 79–109 (1997).
5. V. Tereshchenko, Y. Tereshchenko and D. Kotsur, "Point triangulation using Graham's scan," Fifth International Conference on the Innovative Computing Technology (INTECH 2015), Pontevedra, 2015, pp. 148-151, doi: 10.1109/INTECH.2015.7173370
6. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," ***2012*** Proceedings of the 35th International Convention MIPRO, Opatija, 2012, pp. 1725-1730.