Finger_Count_Recogniser

April 4, 2020

1 Basic Finger Count Recogniser

1.1 Imports/ Dependencies

[1]:
```
import numpy as np
import cv2
import math
```

1.2 Global Variables and Constants

[2]:
```
startrect = (300,300)      #Start point of capturing window
endrect = (100,100)        #End point of capturing window
number_frames = 0          #Frame Count
background = None          #Extracted background from frame
acc_weight = 0.5           #CONSTANT for weighted average
```

1.3 Analyse Background

Analyses the background for first 120 frames and get the average image of the background

[3]:
```
def analyse_background(frame_blurred):
    global acc_weight, background

    if background is None:
        background = frame_blurred.copy().astype("float")

    cv2.accumulateWeighted(frame_blurred, background, acc_weight)
    #Uses passed on frame and background to analyse the background
    #Changes the "background" variable in  each loop
```

1.4 Analyse Hand

Uses the analysed background for subtraction and calculates the contour and its defects for finger count calculation. This method houses the core logic

[4]:
```
def analyse_hand(frame_blurred, img, crop_img):
    global background
    # Calculates the Absolute Difference between the background and the frame
```

```
    diff = cv2.absdiff(background.astype("uint8"), frame_blurred)

    _, thresh = cv2.threshold(diff, 100, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    cv2.imshow('Thresholded', thresh)

    contours, hierarchy = cv2.findContours(thresh.copy(),cv2.RETR_TREE, \
            cv2.CHAIN_APPROX_NONE)

    ################################################
    # Use the contour with maximum size for further #
    # calculations                                 #
    ################################################
    maxi = [cv2.contourArea(cnt) for cnt in contours]
    if len(maxi) == 0:
        cv2.imshow('Finger Counter',img)
        return (diff, 0, None, img)

    maxi = maxi.index(max(maxi))
    cnt = contours[maxi]

    ################################################
    # Find the convex hull and convexity defects of #
    # the contour for analysis                     #
    ################################################
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)

    ################################################
    # Draw the contour and overlay the region of    #
    # interest with the cropped image              #
    ################################################
    crop_img = cv2.drawContours(crop_img, cnt, -1, (0, 0, 255), 2)
    img[endrect[0]:startrect[0], endrect[1]:startrect[1]] = crop_img

    #Set count_defects to zero
    count_defects = 0

    ################################################
    # Checks the different points of convexity      #
    # defects for fingertips                       #
    ################################################
    if defects is None: #To avoid error in case any
        return (diff, count_defects, defects, img)

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

        if angle_between_fingertips_and_fingerdip <= 90:
            count_defects += 1
            cv2.circle(img,far,1,[0,0,255],-1)

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

    x,y,w,h = cv2.boundingRect(cnt)
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

    return (diff, count_defects, defects, img)
```

1.5 Displays the final images

Puts text on images and displays them

[5]:
```
def show_images(diff, count_defects, fin_img, image_without_madness):
    global endrect

    if (diff > 30).any() :
        if(count_defects == 0):
            cv2.putText(fin_img, "Yeah 1 finger !", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)
        elif(count_defects == 4):
            cv2.putText(fin_img, "Yeah 5 fingers !", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)
        elif(count_defects == 3):
            cv2.putText(fin_img, "Yeah 4 fingers !", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)
        elif(count_defects == 2):
            cv2.putText(fin_img, "Yeah 3 fingers !", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)
        elif(count_defects == 1):
            cv2.putText(fin_img, "Yeah 2 fingers !", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)

        cv2.imshow('Finger Counter !', fin_img)
    else:
        cv2.imshow('Finger Counter !', image_without_madness)
```

1.6 Main method

Does the required changes in the original image and calls the required methods with required parameters

[6]:
```
def main():
    cap = cv2.VideoCapture(0) #Turns on web cam and starts capturing
    global endrect, number_frames

    while(True):
        ret, img = cap.read()
        img = cv2.flip(img,1)
        cv2.rectangle(img,startrect,endrect,(0,255,0),0)
        image_without_madness = img.copy()
        crop_img = img[endrect[0]:startrect[0], endrect[1]:startrect[1]]
        grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
        value = (35, 35)
        blurred = cv2.GaussianBlur(grey, value, 0)

        if(number_frames < 120):
            cv2.putText(img, "Wait analysing background", endrect, cv2.
↪FONT_HERSHEY_SIMPLEX, 1, 1)
            cv2.imshow('Finger Counter !',img)
            analyse_background(blurred)
        else :
            diff, count_defects, defects, img = analyse_hand(blurred, img,␣
↪crop_img)
            if defects is None:
                continue
            show_images(diff, count_defects, img, image_without_madness)

        number_frames += 1
        k = cv2.waitKey(1)
        if k == 27:
            break

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

1.7 For calling main() method

[7]:
```
if __name__=='__main__':
    main()
```