Hand Gesture Recognition Using Computer Vision

Presented by

Midhun P (TVE17EC027)

Sreejith S (TVE17EC049)

Vipin Chandran M (TVE17EC061)

Phase 1

Objective –

The intention of phase 1 of our project was to detect a hand (the colour of skin) from an image.

Algorithm –

✓Define range of HSV colour space co-ordinates for skin.
✓Read the image from the file source location.
✓Convert the image from BGR to HSV colour space.
✓Create a mask based on which pixels fall into specified upper and lower ranges.
✓Apply a series of erosions and dilations to the mask.
✓Blur the mask to remove the noise, then apply mask to the frame.
✓Show the skin in the image along with the mask.

Desktop/Untitled Folder/ | Untitled1 - Jupyter Notebook | cv2.error: OpenCV(4.1.0) C:\proje | hand images - Google Search

localhost:8888/notebooks/Desktop/Untitled%20Folder/Untitled1.ipynb?kernel_name=python3

jupyter Untitled1 Last Checkpoint: 10 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [1]:
```
import cv2
import numpy as np
lower=np.array([0,48,80],dtype="uint8")
upper=np.array([20,255,255],dtype="uint8")
```

In [6]:
```
frame=cv2.imread(r'C:\Users\sreej\Desktop\Untitled Folder\Hand.jpg')
converted=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
cv2.imwrite('./1.png',converted)
skinMask=cv2.inRange(converted,lower,upper)
cv2.imwrite('./2.png',skinMask)
kernal=cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(11,11))
cv2.imwrite('./3.png',skinMask)
skinMask=cv2.erode(skinMask,kernal,iterations=2)
cv2.imwrite('./4.png',skinMask)
skinMask=cv2.dilate(skinMask,kernal,iterations=2)
cv2.imwrite('./5.png',skinMask)
skinMask=cv2.GaussianBlur(skinMask,(3,3),0)
cv2.imwrite('./6.png',skinMask)
skin=cv2.bitwise_and(frame,frame,mask=skinMask)
cv2.imwrite('./final.png',skin)
```

Out[6]: True

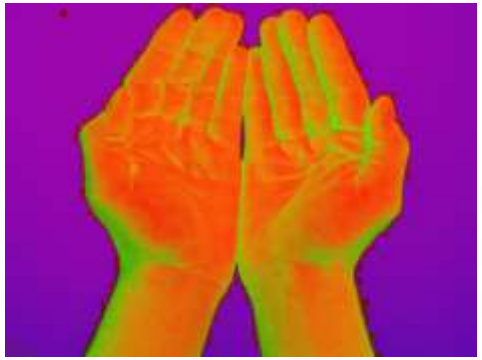
In [ ]:

Type here to search

ENG 01:23 23-03-2020

Phase 1 results

Hand.jpg (Input image)



1.png

2.png

3.png

4.png

5.png

6.png

final.png (Output image)

Phase 2

Objective –

The purpose of phase 2 of our project is to recognize different hand gestures from a live video.

Algorithm – (only additions to phase 1 of the project are mentioned below)

- ✓Open camera and capture frames from it.
- ✓Create a binary image where white will be skin colours and the rest is black.
- ✓Find contour with maximum area and create a bounding rectangle around that contour.
- ✓Find convex hull.
- ✓Use mathematical calculations to find the number of convexity defects.
- ✓Show required images and print number of fingers.
- ✓Close the camera if 'q' is pressed.

Desktop/Jupyter/ | Test - Jupyter Notebook

localhost:8888/notebooks/Desktop/Jupyter/Test.ipynb

jupyter **Test** Last Checkpoint: 8 hours ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [2]:

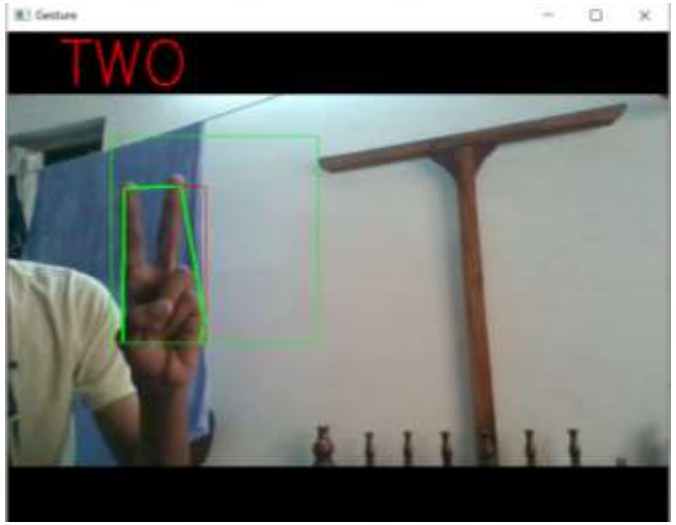
```
import numpy as np
import cv2
import math

capture = cv2.VideoCapture(0)

while capture.isOpened():
    ret, frame = capture.read()
    cv2.rectangle(frame, (100, 100), (300, 300), (0, 255, 0), 0)
    crop_image = frame[100:300, 100:300]

    blur = cv2.GaussianBlur(crop_image, (3, 3), 0)
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
    mask2 = cv2.inRange(hsv, np.array([2, 0, 0]), np.array([20, 255, 255]))
    kernel = np.ones((5, 5))
    dilation = cv2.dilate(mask2, kernel, iterations=1)
    erosion = cv2.erode(dilation, kernel, iterations=1)
    filtered = cv2.GaussianBlur(erosion, (3, 3), 0)
    ret, thresh = cv2.threshold(filtered, 127, 255, 0)
    cv2.imshow("Thresholded", thresh)

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    try:
        contour = max(contours, key=lambda x: cv2.contourArea(x))
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(crop_image, (x, y), (x + w, y + h), (0, 0, 255), 0)

        hull = cv2.convexHull(contour)
        drawing = np.zeros(crop_image.shape, np.uint8)
        cv2.drawContours(drawing, [contour], -1, (0, 255, 0), 0)
        cv2.drawContours(drawing, [hull], -1, (0, 0, 255), 0)

        hull = cv2.convexHull(contour, returnPoints=False)
        defects = cv2.convexityDefects(contour, hull)
```

Type here to search

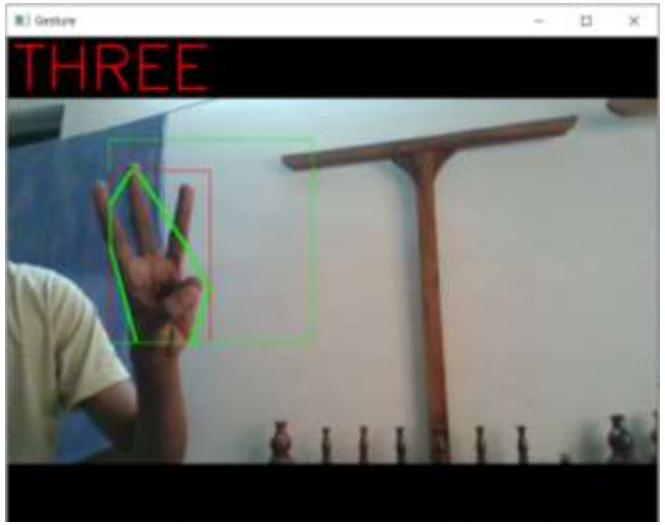
ENG 17:16 24-03-2020

Phase 2 results

Gesture 1

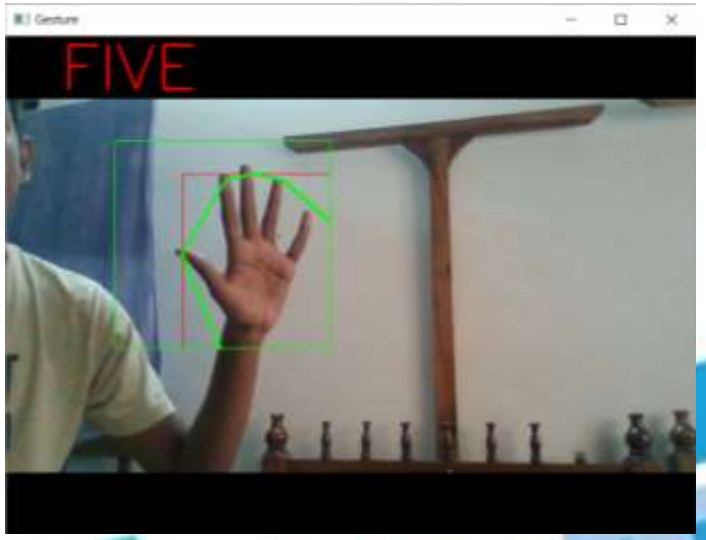


Gesture 2



Gesture 3

Gesture 4



Gesture 5


Thank
you!