Rochester Institute of Technology

NSSA 102

Computer System Concepts

Basic Hand Gesture Recognition Using OpenCV on a Raspberry Pi 3

By

Jeremy Peters

Rochester Institute of Technology

Introduction **3**

Structure **3**

Code **4**

Bugs **6**

Sources & References **7**

Rochester Institute of Technology

Introduction

The purpose of this project was to isolate a single hand raised (palm facing the camera), determine the number of fingers being held up, and then light up a series of LEDs to indicate how many raised fingers the program detects. To accomplish this, the open source library OpenCV was used to process a real-time video feed on a Raspberry Pi 3 (RasPi).

I've had previous experience working on OpenCV on the same RasPi, which is why I chose this as my project. My last project using OpenCv on the RasPi was an attempt to isolate a human face and track pupil movement. This project had several challenges resulting from performance issues. These issues were due to the need for high frame-rate, high resolution video to properly track pupil movement at a reasonable distance (Approximately 12 -18 Inches).

Based off this knowledge, I decided that it would be much more feasible to track hand movement as it wouldn't require a high frame-rate and/or high resolution video feed. After conducting some research and a proof of concept, I began work on the project.

Structure

This project is split into two main components: program and hardware. The program is where the majority of my time spent working on this project went. It took approximately eight hours, over several days, to complete the program. The hardware portion of this project was quite simple and took less than thirty minutes to implement.

Code

The code I created for this project follows a linear path in its progression of processing the supplied video feed. The video processing is contained within a single loop that can be broken if the Esc key is pressed.

The loop begins with minor image adjustments such as flipping the image, adjusting the HSV (Hue, Saturation, Value) thresholds, and applying a slight blur to reduce sharpness that could hinder edge detection. From there, the program will try to create a contour around the isolated area, in this case, an up-raised hand. The program than checks to see if the area contained within said contour meets a minimum area threshold. If this condition is met, the program will then try to calculate a set of indices that make up what is known as a convex hull.

Using these indices, the program then looks for defects between the original contour and the convex hull it had calculated. These defects are then fed into a clever bit of math (Fig. 1) that looks to find the angle residing at the farthest point away from the hull. This is the angle that is formed between two fingers such as when making a peace sign. If the calculated angle is less than 90°, then the program will assume it is an angle between two fingers and add it’s occurrence to an ongoing count.

Finally, after the program has finished counting the angles, it will light up a series of LEDs to show how many fingers it thinks are being held up.

```
start, end, far = tuple(contour[ds][0]), tuple(contour[de][0]), tuple(contour[df][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 = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c))
```

Figure 1

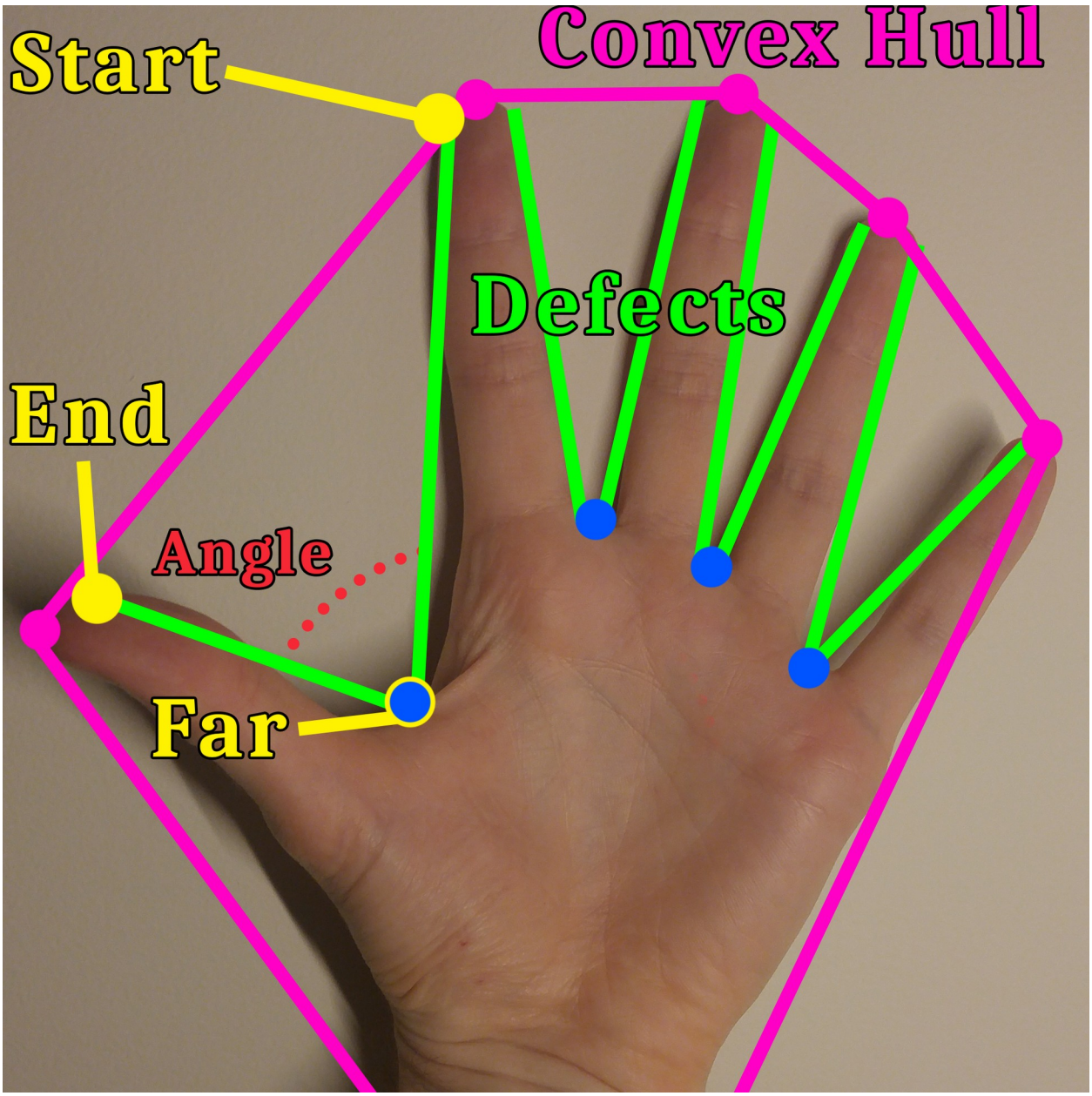

Figure 2

Bugs

During the course of coding this programming, I was plagued by a frustrating bug. When executing the following line of code there is a chance that a bug can occur.

```
cv2.convexHull(contour, returnPoints=False)
```

This bug involves how OpenCV returns indices when calculating a convex hull. It is unclear what causes it, but OpenCV will sometimes return a set of indices for a contour that contains a self-intersection (Fig 3.). When this set of indices is then passed into the function for detecting defects, it causes an error which exits the program that called said function. This bug is still present in the final iteration of the program. To circumvent the constant program exits due to errors, the function in which this bug is present is called from a “Try” statement which simply ignores the errors produced. From my research of the bug, I was unable to conclude if there was an actual fix beyond what I’d implemented.

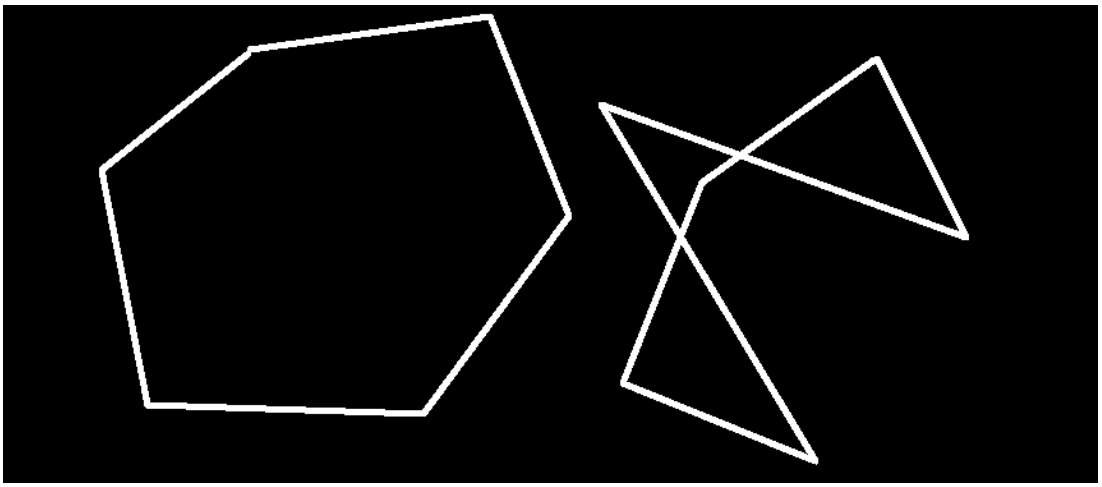

Figure 3
Regular Contour (left), Self-Intersecting Contour (right)

Sources & References

My Code
https://github.com/JeremySPeters/Hand-Tracker/blob/master/Hand_Tracker.py

Reference Code
https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python

Thread on openCV bug
https://github.com/opencv/opencv/issues/4539

Convex Hulls Explained
https://www.youtube.com/watch?v=VP9ylElm1yY