Open Gesture Report

Eshan Ramesh Gatlen Culp

2020 July 24th

1 Introduction

1.1 Technological Background

With more and more powerful devices and algorithms, computer vision is more popular than it ever has been. We see it in our computers and camera all the time. It's the reason for snapchat filters, object detection in self driving cars, and artificial intelligence.

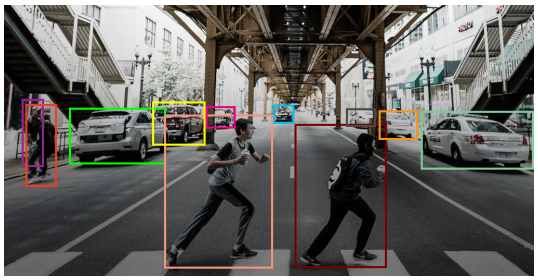

Figure 1: Camera Vision

1.2 Motivation

Mainly Educational

We wanted to expand our knowledge about camera vision through the creation of this project. We saw that there was a huge opportunity to control computer devices with non-conventional controls (ie: cameras). These devices might be televisions, computers, smart home devices, or even robots. We decided to take up the task of creating a basic input simulation program to learn the basics of computer vision.

1.3 Project Description

Runs a customizable keyboard or mouse input command based on the number of fingers you are holding up. For this presentation we use the following set up. For our project we created a program that runs a customizable keyboard or mouse input command based on the number of fingers you are holding up. For this presentation we use the following set up.

Fingers Held Up	**Command**
1	None
2	Down
3	Up
4	None
5	None

1.4 Practical Applications

1. Presentations. As the project currently stands, we can see it helping out people during a presentation, in which the speaker could make hand gestures to move between slides instead of relying on an expensive clicker or a second person to change slides.
2. Media player. It could also be adapted into a media player, gesturing for music to start or stop, to raise or lower the volume, skip songs, etc.
3. Remote interaction This program may also be used to assist anyone remotely interact with their computer for any other reason we have not listed here.
4. We will discuss the future possibilities of this application later on.

2 Methods

2.1 Original Algorithm

We use an existing algorithm as a base, published by Ilango which is an adaption of the algorithm suggested by Malima, Ozgur, and Cetin and make improvements to it.

The algorithm is intended to be computationally fast and used in robots.

2.1.1 Preprocessing

It's necessary to extract and normalize hand images. A small "region of interest"(ROI) is first selected from the frame, and is then:

1. Resized to 700x700px
2. Mirrored horizontally
3. Converted to grayscale
4. Gaussian blurred with kernel size (7,7) to smooth rough edges.

2.1.2 Hand detection

We use the first 50 frames of the program to compute the "running average" for each pixel of the background.

Now, we count a pixel $I_{i,j}$ as part of the hand $\iff$ $I_{i,j} > t$ where t is some threshold.

2.1.3 Palm centering

1. Find the contour of the hand
2. Compute the convex hull of the hand contour
3. Find the extrema of the convex hull
4. Compute the average of the convex hull and assume it to be center of the palm

2.1.4 Finger counting

1. Create a "palm circle" with radius 0.8 times the maximum euclidean distance from the center of the palm to any extrema on the convex hull.
2. Compute the intersections of the palm circle and the original thresholded image using a simple bitwise AND. The remaining binary image contains arc contours for each of the fingers
3. A remaining contour is counted as a finger $\iff$ the contour

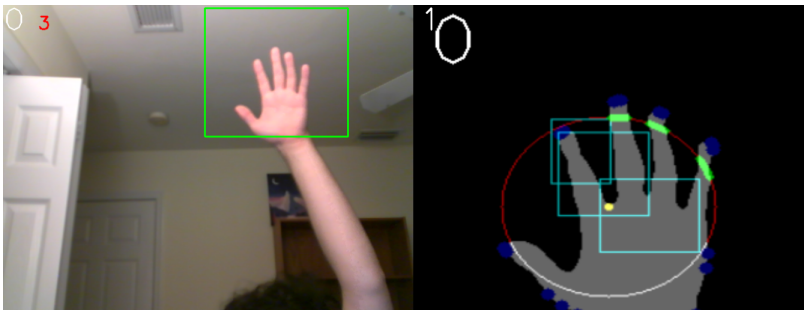

Figure 2: Original algorithm debugging image

- Is not the wrist (not below the palm)
- Does not contain a number of points that exceeds 25% of the palm region's circumference.

2.1.5 Demonstration and problems

As you can see in figure 2, there are several problems:

- Circle goes above pointer
- Palm center is too high
- Thumb is counted as part of the wrist.
- Count fluctuates wildly: no good for computer control

2.2 Improved Algorithm

Original	**Improved**
Full circle	Upper segment of an ellipse.
Used midpoint between extrema of the convex hull enclosing the hand.	Used the moment (weighted average) of the thresholded image to find palm.
No fluctuation/transition accounting	Using time the fingers are consecutively held in frame to account for fluctuations.
N/A	Gesture based commands which are read in from a YAML file at the beginning of the program.
N/A	Hackable NumberRecognition class that can be inherited from and altered.
N/A	Debug display to show how the numbers are visually counted.

You can see a demonstration of this improve algorithm in figure 3.

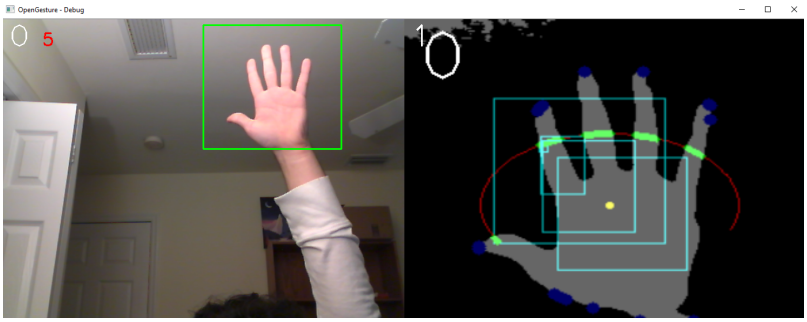

Figure 3: Optimized algorithm debugging image

3 Results

3.1 Issues and Future Changes

1. Method to allow for different hand orientations. Currently, rotating the hand to point any direction other than directly up causes the program to fail to recognize the fingers.

2. Real time hand tracking. Currently you need to place your hand into a single square in a background that is not changing.

3. Accommodating for different lighting situations and background removal. Currently, backgrounds and lighting other than white can mess up the readings.

4. Simulating analog input such as moving your hand in and out/up and down/left and right to move the mouse. Our program simply does not have this feature.

5. Automatic command profile switching based on application. (ex: certain commands for photoshop, another set of commands for a presentation) Currently our program only has a static set of commands.

3.2 Future Applications

1. Augmented Reality (AR) or Virutal Reality (VR) (Natural 3D tools)

2. Smart home controls.

3. Robot manipulation.

3.3 Get your copy!

OpenGesture is very easy to install!

Figure 4: Oculus Quest Using Hand Tracking

Install it with one command: "pip install OpenGesture"
And run it with one command too: "opengesture"
Usage options:

1. opengesture -t [threshold]
2. opengesture -c [camera index]
3. opengesture -s [path/to/settings.yaml]
4. opengesture –help

References

[1] G. Ilango. (Apr. 6, 2017). Hand gesture recognition using python and OpenCV, Gogul Ilango. Library Catalog: gogul.dev, [Online]. Available: `https://gogul.dev/software/hand-gesture-recognition-p1` (visited on 07/24/2020).

[2] A. Malima, E. Ozgur, and M. Cetin, "A fast algorithm for vision-based hand gesture recognition for robot control," in *2006 IEEE 14th Signal Processing and Communications Applications*, Antalya, Turkey: IEEE, 2006, pp. 1–4, ISBN: 978-1-4244-0238-0. DOI: `10.1109/SIU.2006.1659822`. [Online]. Available: `http://ieeexplore.ieee.org/document/1659822/` (visited on 07/17/2020).