**Finger/Hand Cursor on MATLAB**

By: Jae Woo Ok

**Personal Views:**

The field of computer vision is ubiquitous in our modern-day life, spanning from popular media to film, games, self-driving cars, and even the way we unlock our phones. For instance, computer vision is used in motion capture technology which is useful for 3D animation, but also useful for modeling in the medical field. Self-driving technology is largely based on computer vision because it requires the car to see where it is going and what is on the road to prevent accidents and accurately drive from one location to another. Facial recognition technology is used for security in homes, phones, and even in some security camera footage to prevent intruders. In general, anything that we can do with our eyes, we can seek to automate. More specifically, computer vision tasks include different methods of acquiring, processing, analyzing, and understanding digital images to elicit appropriate actions or decisions.

In the course, we (as a class) have learned many technical skills and background knowledge about computer vision. First, we started with CCL, or Connected Component Labeling, which is a way in computer vision to detect regions that are connected, and also label the different component regions. Second, we learned about morphological operators which help remove imperfections of binary images by accounting for the form and structure of the image. Third, we learned about histogram equalization, which is a method that uses a histogram to accomplish contrast adjustment. Fourth, we used what we learned in histogram equalization for a technique called color segmentation. This technique was very important for this project because it would allow separation between foreground and background, allowing accurate tracking of the hand in the foreground. Then, we learned about the Canny Edge algorithm and the Hough transform to track the edges of a digital image, and also extract the most significant edges of the image. Finally, we used motion tracking with bounding boxes to track objects in the frames of a video.

In the future, I want to study more about some optimization techniques of some of the implementations and algorithms that we have learned in class. Optimization is a very important aspect of design as it would take into account variables such as efficiency, speed, and memory/area/space (of hardware). There are also other very interesting computer vision algorithms that I would be interested in learning, because that would probably be different applications of some of the concepts we have learned in class. For instance, there is the Viola-Jones algorithm which is the first object detection framework to provide competitive object detection rates in real-time.

**Project description**

**Problem statement:**

In the field of gesture recognition and image processing, finger tracking is a high-resolution technique that is employed to know the consecutive position of the finger or fingers of the user. Finger tracking is important for communication between humans and computers to be more intuitive through hand gestures and movements. This kind of technology can be expanded so that in the future, there might be technological devices that can be controlled only through hand and body gestures. There are many ways to implement finger/hand tracking, but in this project and for the sake of the class, an optical motion capture system will be implemented through MATLAB.

**Project goal:**

The goal of the project was to use the techniques we have learned in class and in the homework to implement a finger/hand tracker. More specifically, to track the movement of fingers and hands so that it would correspond to the movement of the mouse cursor. For the code, instead of using the given video for analysis, the goal was to open the webcam to track the finger in real time.

**My design**

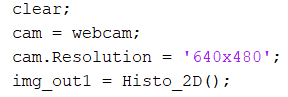
In this section, I will provide a rundown of the algorithm and the code of the MATLAB program. The general gist of the program is to draw a bounding box for each frame around my hand, and then use the position of that bounding box to control the position of the mouse cursor. Instead of analyzing an existing video, the code opens up the webcam and uses the webcam instead.

NOTE: It will be obvious from the explanation below, but my code works best if:

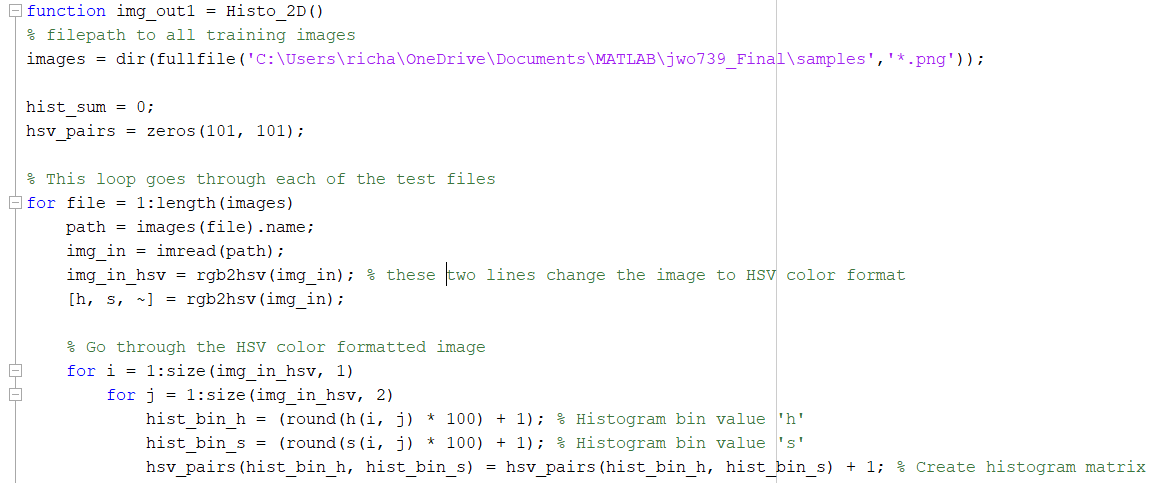
1. The hand is the largest bounding box in the image
2. You are wearing a long-sleeved shirt (to isolate just your hand)
3. There is good lighting and it doesn’t interfere with color segmentation

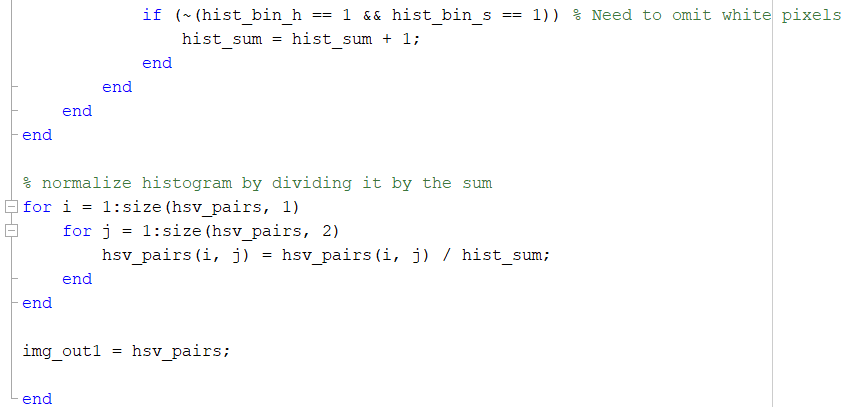
The main code is in ‘fingercursor.m’. The images that I have used for the project is in the ‘images’ folder, and the sample images used for the histogram is in the ‘samples’ folder. The code for the histogram is in ‘Histo\_2D.m’ and the code for color segmentation is in ‘Color\_Seg.m’. In each section below, I will first post the code that was used and then proceed to explain the code.

Here is the start of ‘fingercursor.m’:



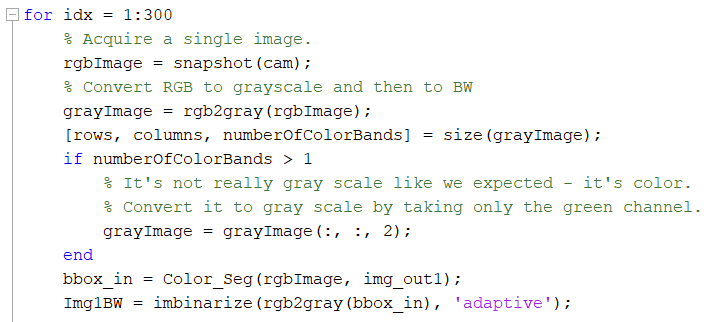
Code of ‘Histo\_2D.m’ (code of the 2D histogram):



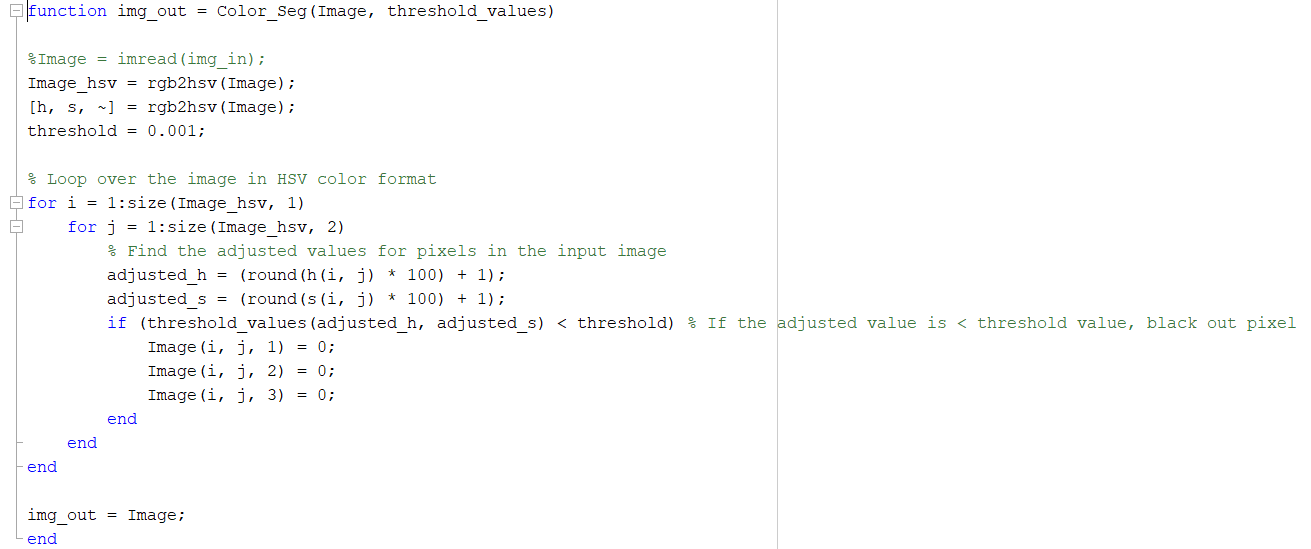


The program starts by opening the webcam and setting its resolution to 640x480. The webcam I used was the ‘integrated webcam’ in my laptop (Model Dell XPS15) and so I picked the resolution that best fit my webcam screen. Next, I took two pictures of my hand in a gun-shape and turned them black and white using ‘rgb2bw.m’. These two pictures were used in my 2D-Histogram program (used in MP4) and the output was stored in the variable ‘img\_out1’.

Here is the next part of ‘fingercursor.m’:



Code for color segmentation, ‘Color\_Seg.m’:



‘for idx = 1:300’ is the start of my main for-loop. 1:300 means that it will go through 300 frames of streamed video. Each iteration of the loop goes through one frame of the video. Each frame was first converted into grayscale. If there was more than 1 color band of the grayscale image, I made sure to only take the green channel by using ‘grayImage = grayImage(:, :, 2)’. Then, I used my color segmentation program (‘Color\_Seg’) and the output of the 2D-Histogram and turned the image into a completely black and white image by using the built in MATLAB function ‘imbinarize’.

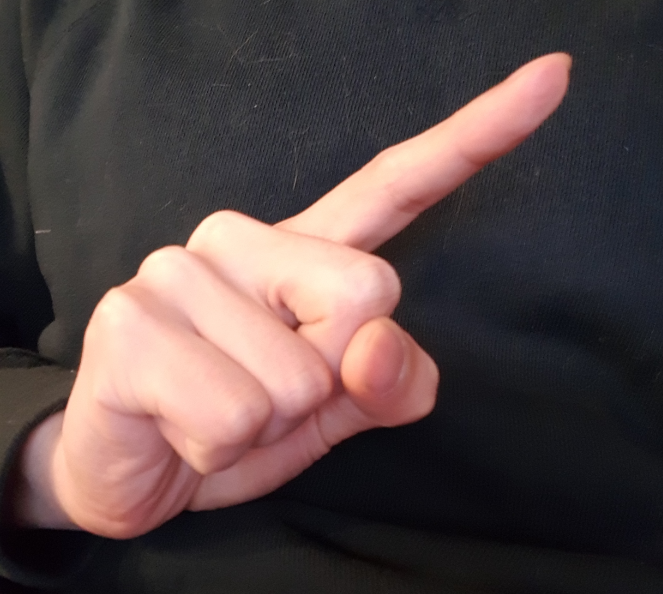
Final part of ‘fingercursor.m’:



Now I needed a way to actually draw the bounding boxes. This was done using the built in MATLAB function ‘regionprops(img, parameter)’. The input image had to be a logical black and white image, so I used ‘logical(Img1BW)’. The parameters that I need from regionprops were the ‘BoundingBox’ and ‘Area’ parameters. In order to just get the largest bounding box and remove all the other smaller ones, I put the areas of all the bounding boxes into an array, and sorted them in descending order. In this array, the largest area is the first value stored in the array. Then, if the area of the given bounding box is the largest one, a red rectangle is drawn using the x, y, width, and height values of the bounding box. The built in MATLAB function ‘set(0,’PointerLocation’,[x, y])’ was then used to set the position of the mouse cursor at the given [x, y] position. For this project, I set the [x, y] position as x= -3\*boundingBox\_xpos+1000 and y=-3\*boundingBox\_ypos+1000 so that the mouse cursor follows the position of the bounding box, and also so that the mouse appears around the location of the video that appears on the screen. Finally, everything was drawn and the camera variable was cleared to turn off the webcam of my laptop.

**Results and Analysis**

Here are the results of each step of the process. The video of my program working is in ‘me\_using\_the\_program.mp4’ where my friend helps me record me using my hand to control the mouse cursor. Note, for some reason, the video only opens with Windows Media Player on my computer. Also, the webcam of my computer is located right above my keyboard and below the screen, so the angle of me on the screen looks a little strange. As mentioned above in the design portion, there are some requirements for the code to work properly.



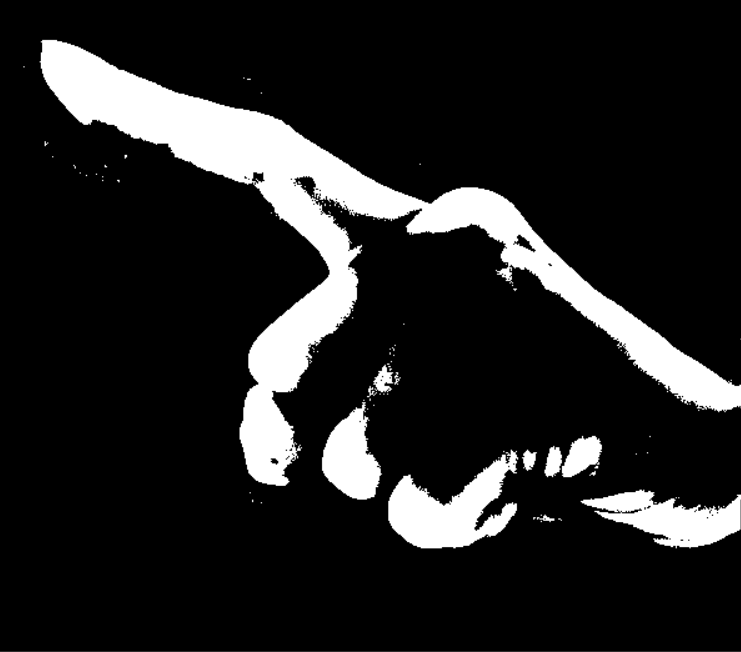
*Figure 1: Original Image of Hand Sample 1*



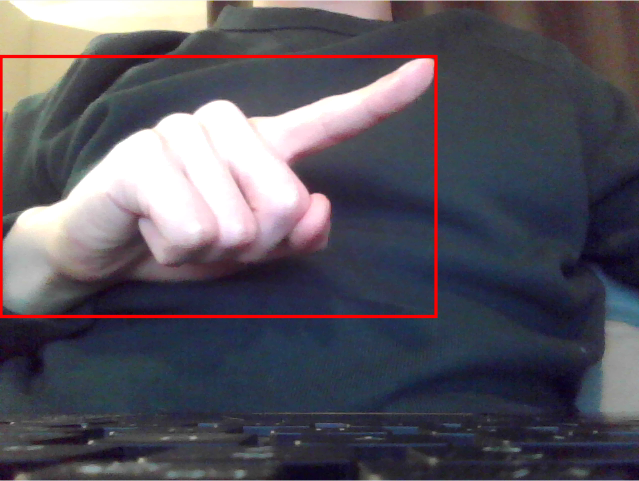
*Figure 2: Original Image of Hand Sample 2*



*Figure 3: Image of Black and White Hand Sample 1*



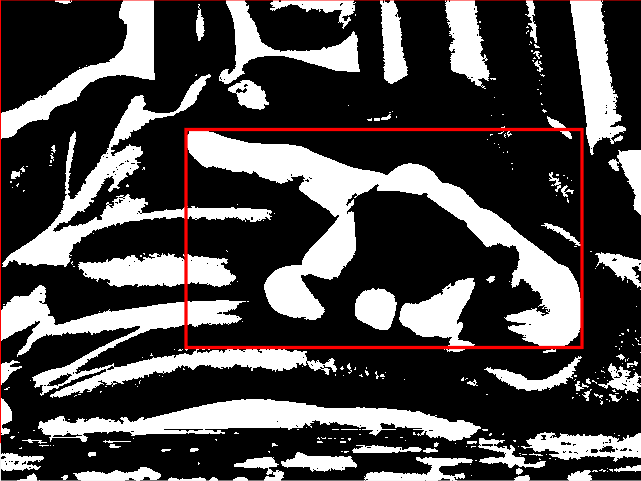
*Figure 3: Image of Black and White Hand Sample 2*



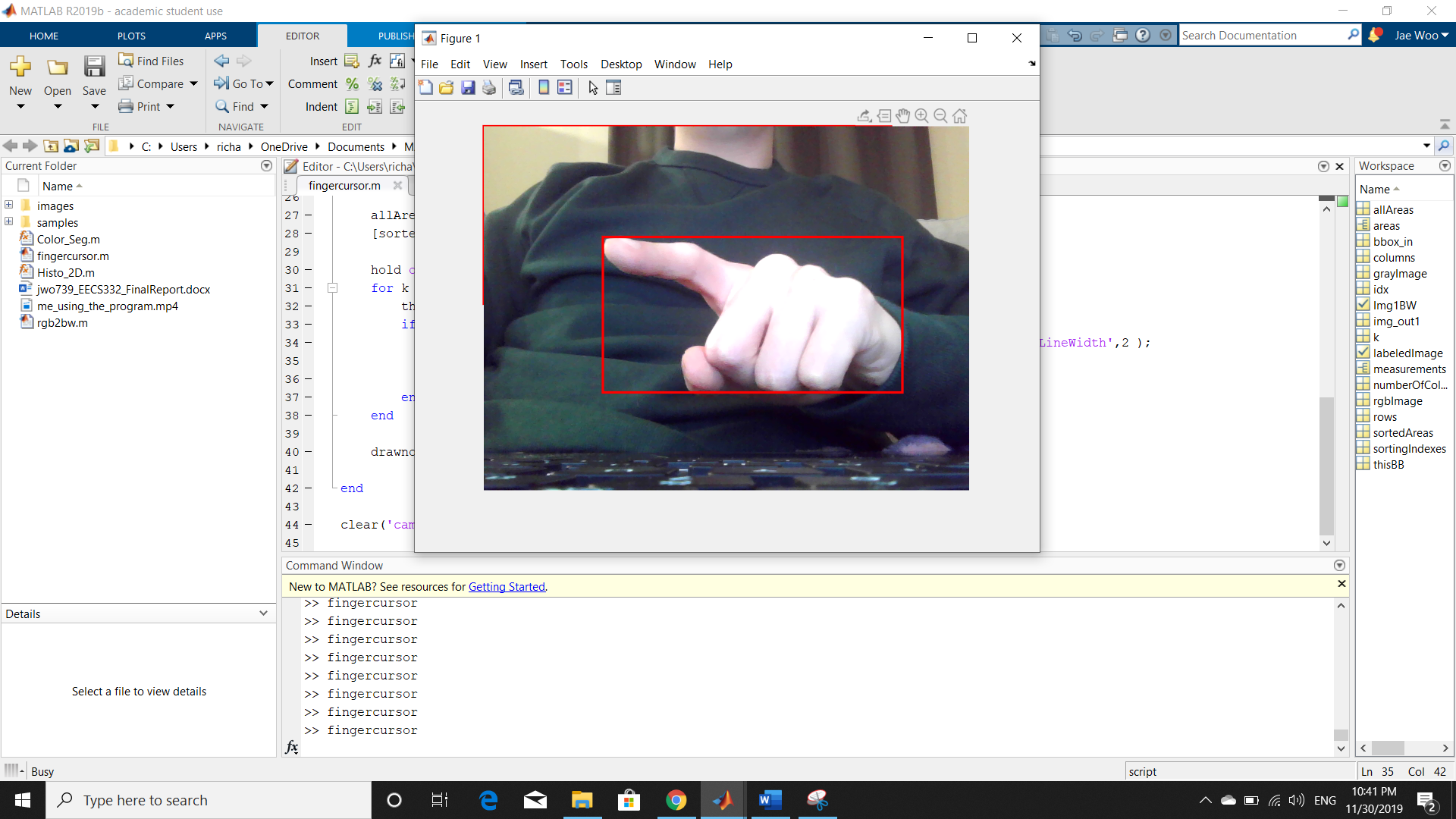
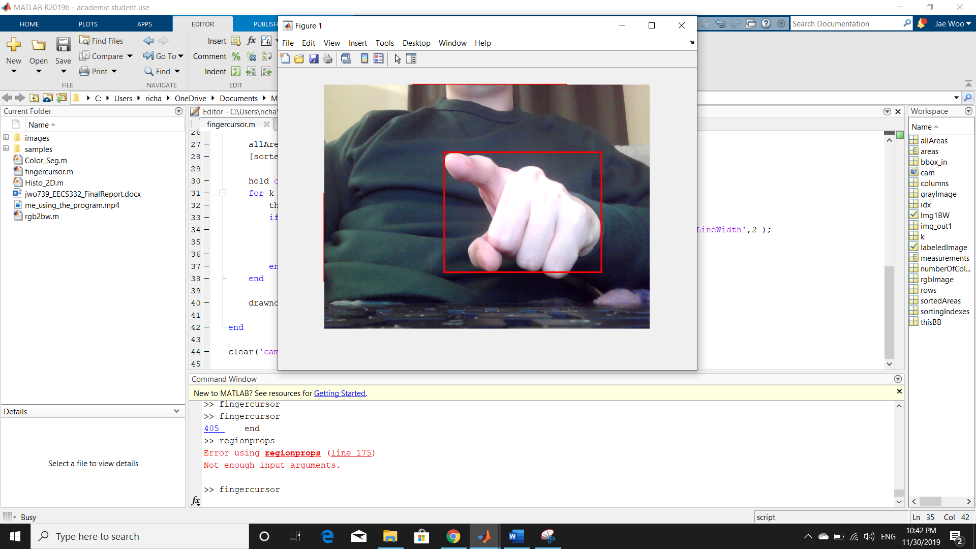
*Figure 4: Image of Bounding Box Around Hand*



*Figure 6: Image of Bounding Box Around Hand 1 (Black and White)*

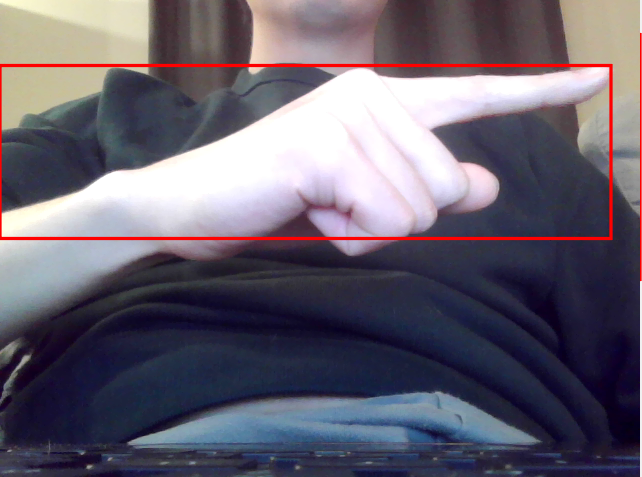


*Figure 7: Image of Bounding Box Around Hand 2 (Black and White)*



*Figure 8: Image of code working on my hand. Bounding box is tracking my hand and it is controlling my mouse cursor*

*\*\* Note: The mouse does not show up when I use the ‘printscreen’ function, so in order to see the mouse movement clearly, the video is much better.*



*Figure 9: Issue of not wearing a long-sleeved shirt as it does not isolate the hand for the correct bounding box*



*Figure 10: Issue of the hand not being the largest bounding box in the image, which isolates the head, which draws the bounding box on the head and not the hand*

For each frame, the bounding box is drawn around the hand and by tracking the hand using the bounding box, the mouse cursor is moved depending on the position of the bounding box. From the result, it can be seen that the mouse cursor does move with the movement of the hand pretty well. However, the algorithm is not perfect. For instance, sometimes if the background is a very similar color to the hand, or there is ambiguous lighting, the bounding box algorithm detects and draws the bounding box in the wrong position, causing the mouse cursor to move erratically. Another observable issue is that the mouse moves opposite of the camera, but I think this is because my webcam inverts pictures and videos.

**Remarks and future work**

After running the program, the way the mouse cursor responds to the movement of my hand is pretty good. However, with further experimentation, there can definitely be some improvements to my program. Sometimes, if the background is a very similar color to the hand, or there is some ambiguous lighting, my detection becomes confused. Some examples of this is if there is a head that is included in the video, or if the light is too yellow/red etc. Also, the algorithm would work better if it somehow managed to keep the mouse cursor steadier as it moved, which correlates to the steady area of the bounding box. I also would like to figure out a way to more clearly isolate the bounding box around the hand, even with the interference of a sleeveless shirt or with other skin colored objects in the picture. Perhaps in the future, I can also add some additional functions, such as a mouse-click with a different gesture.

For some additional topics, I feel like some optimization techniques for some of the algorithms learned in class would be interesting. Also, maybe some topics on how to connect different gestures to different functions would be useful, as it not only needs to use the image tools library, but also the Java-Bot library and functions. Finally, maybe some other significant image analyzing algorithms other than the Canny-Edge or the Hough Transform might be interesting as well.

**Course feedback and suggestions**

The course has been very informative and I have learned a lot about technical skills and background information about some of the techniques used in computer vision. The structure of the class seemed to be at a good pace. The professor gave us enough background knowledge and information to get started on the code, and we could figure out what to do based on what we learned. The projects cover the material learned in class, and it was very helpful to do the homework because we could implement the algorithms and see the effect with our own eyes. It was a fair assessment of the knowledge because the algorithm was taught in class, but the implementation was largely up to us, in whatever coding language that we wanted to go for. I personally went for MATLAB because of the amount of built in matrix manipulation functions MATLAB contains.

However, I do wish that we did go over more example code samples to further help us with the implementation. Especially for students who are not CS students, or students who need some review on some languages that would be useful for this class, such as MATLAB or Python.

The class has made me wonder about some items that I use and see in my everyday life and I feel like I am starting to understand more about the world of computer vision around me. Such as facial recognition for the phone, camera focusing on the face, and automated driving on highways for some cars. On the other hand, it also makes me realize how vast and deep the field is and it is really mind boggling to wonder about the incredible discoveries and inventions that are being made today in the field.

For people who are wondering about taking this class or not, I feel like this is definitely one of the more interesting CS courses I have taken at Northwestern, and would recommend it to anyone who is even remotely interested in the field of computer vision. As I am thinking of future classes, I am personally thinking of taking some more computer vision classes in the future, such as the ones focused on optimization.