Xiaowei Yuan

ECE 332

06 December 2022

ECE332 FINAL PROJECT REPORT

FINGER CURSOR

CONTENT

ABSTRACT

In this Final project, I implement a program called Finger Cursor, which is s simple way to trace a fingertip and transfer the moving of fingertip into lines in computer windows. In this project, I combined some techniques which were taught in lecture and used the external API such as MediaPipe. The result turns out to be basically functinable. Drawbacks also exist, more improvements are supposed to be added if one more month was given.

INTRODUCTION

Nowadays, real-time fingertip tracer detection is still a challenge in Computer Vision and HMI fields, due to interference by camera, change of light and immature fingertip tracing methodology. At the same time, treating the fingertip as a virtual mouse is a popular method since it connects humans and computers without any physical devices. It helps a lot in below situations:

1. When professors take lectures on the stage, they focus on interaction with students and it's inconvenient for them to stand by the computer and do handwrite. Finger cursor helps to boost the lecture efficiency. The same idea applies to any presentations or speeches on the stage.
2. If one's aim gets injured and can only move fingers, it's a useful way to do handwriting and hand gestures.
3. As more computer interactive games emerge, finger cursor or tracer can serve as an interactive command in the games which can be captured by the computer cameras.

Above are some usage of finger cursor, but more can be developed if mature systems come out.

GOAL

1.The camera can be successfully turned on and off.

2.The Video to Image and Image to Video translation is smooth. Each frame is successfully extracted.

3.The program can trace the fingertip of the index finger in each frame and shows a bounding box around it.

4.The track of index fingertips is marked as a smooth tracking line.

5. The program can recognize two different gestures: pointer and gun, and apply different colors of lines on them.

6. Tracking line disappears when the length of the line is too long.

APPROACH

Camera set up and configuration set up

In this step, I do the preparation work for the finger cursor project and build configurations. To be first, use `VideoCapture.()` to activate the computer camera. In the test case, I use a computer camera (which is also `device(0)`). If the external camera is connected, the devices can be edited by the user themself.

Then the image width and length are extracted and a related kernel is built based on the pixel information. The similar code can be found in the MP4 and I've used the similar methodology to build the kernel in order to benefit later pixel training. The color of skin and color of the bounding box and lines are all settled in this step. Two gesture images are initialized. Gesture images are real photos of my own hand. I use these two pictures and name them after

'pointer' and 'gun'. In this step I set the kernel filter size and gesture filter size to be 6 and the gesture filter threshold to be 0.7. It's the most fitted size and threshold. Kernel size to be 10*10.

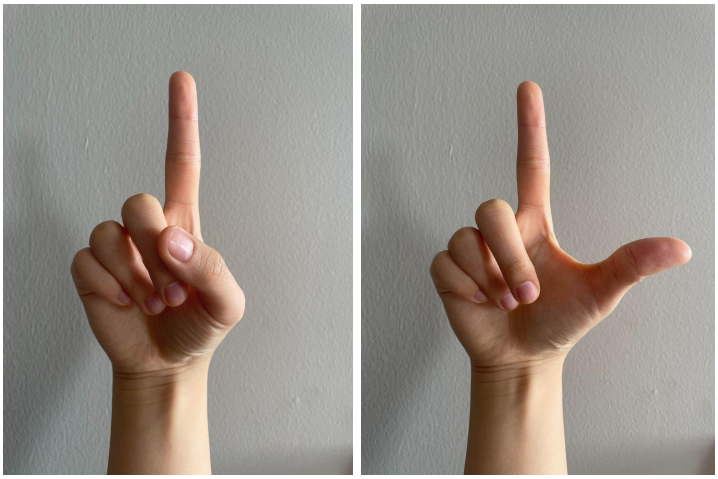

Fig Above: 'pointer' and 'gun', serves as an input gesture, a photo of my own hand.

Generate a loop that can extract the frame from computer camera

This is a simple way to achieve Video to Image and Image to Video translation. The similar methodology can be found in the lecture, and I simply copy and paste the code from MP7 to this project.

In detail, I generate a loop to detect if the camera is on or off:

If YES:

Read each frame using `cam.read()` and do the following several steps.

If No:

Go straight to the last step.

I designed a function which could break out the camera with pressing the keyboard 'P'. If the user presses P and tries to turn off the camera, the loop will immediately stop since the speed of reading each frame is 30fps.

Find the fingertip of the index finger

In this section, I firstly use the strategy we've learned from class to isolate our hand from the image.

According to what I've learned in the MP4, there are three kinds of ways to isolate the hand: RGB, N-RGB and HSI. Instead of using the mouse drawing function to determine the area of the hand, I directly use skin color RGB, there's a min color RGB and max color RGB, any pixel with the RGB inside this range can be counted as skin color.

I've tested three kinds of methodology and the results show that HSI performs the best. That's the reason I chose HSI as the method in my hand segmentation function.

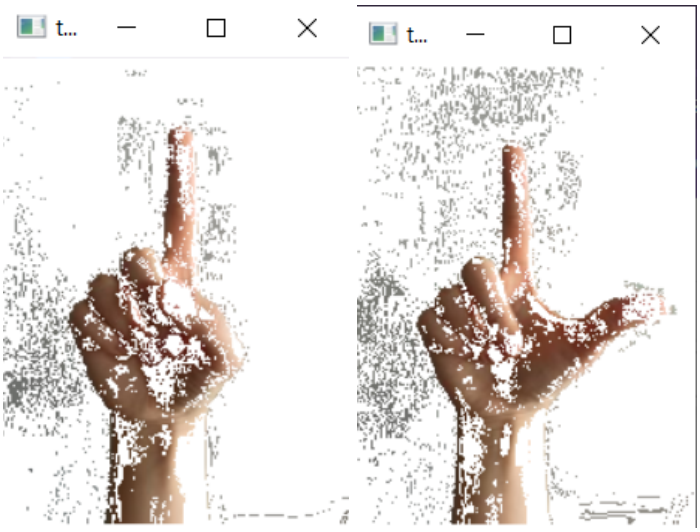

Fig Above:hand segmentation result using N-RGB method, original image is shown in the 'Camera set up and configuration set up' part named 'pointer' and 'gun'

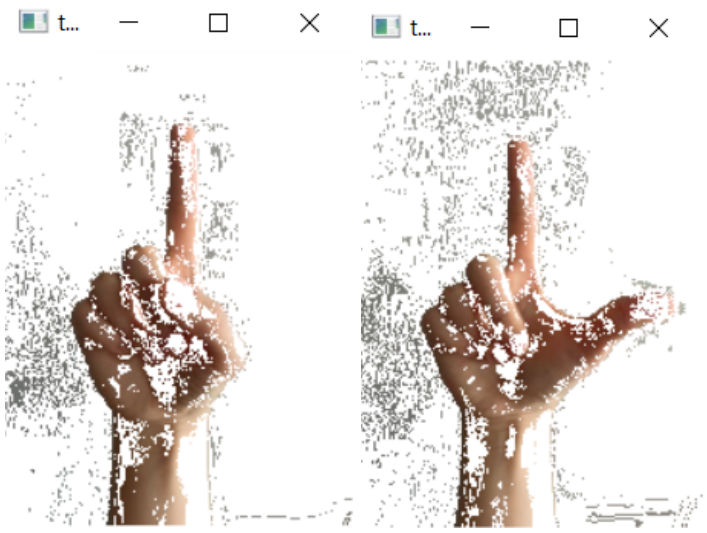

Fig Above:hand segmentation result using RGB method, original image is shown in the 'Camera set up and configuration set up' part named 'pointer' and 'gun'

Fig Above:hand segmentation result using HSI method, original image is shown in the 'Camera set up and configuration set up' part named 'pointer' and 'gun'. The results are the most clear images compared with the other two methods.

A change and upgrade of the isolation methodology is made according to the paper ‘ Real-time virtual mouse system using RGB-D images and fingertip detection’[1]. There I used RGB-D images and fingertip detection, where the hand region of interest and the center of the palm are first extracted using in-depth skeleton-joint information images.

After the hand isolation, I use the MediaPipe API to track the contour of the hand. MediaPipe is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame.

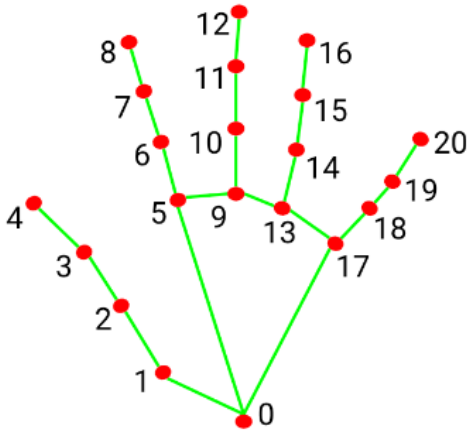

Fig Above: hand landmarks using MediaPipe API, the 20 points with red dots are finger joints of a hand

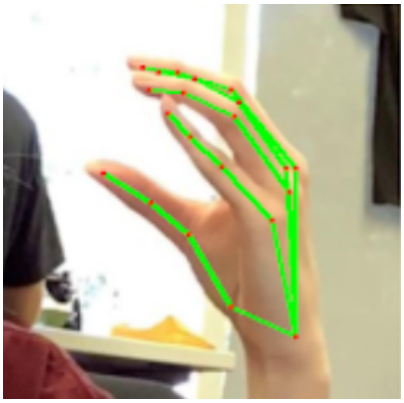

Fig Above: example of Aligned hand crops

As shown in the image above, we can easily find the joints of the hand using the MediaPipe API hand track part. And in order to trace the index fingertip, we need to apply the hand isolation segment into aligned hand crops and find the 8th point of that hand.

Then we generate a circle bounding box around that point, and this bounding box will be shown to the user when doing the finger cursor.

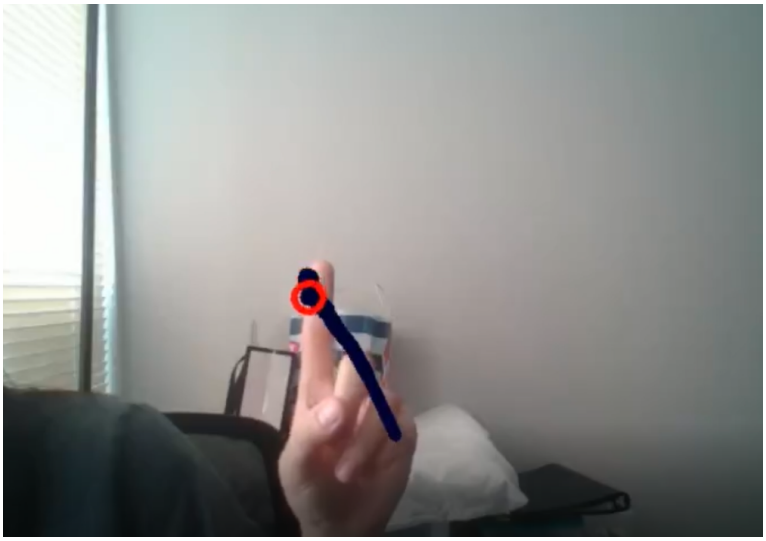

Fig Above: red circle is the bounding box of index fingertip, captured in a random frame of my demonstrate video

Recognize the hand gesture and match up

In this step, i firstly set up some hand gestures as I'vr mentioned before:

1. index fingertip up – point

   In this pointer gesture, I calculate the similarity of my gesture image and the hand I've extracted from the camera. If the most area is covered, then the hand can be decided to be a gesture 'pointer' and lead into the 'pointer' function. There the color of the bounding box will be red while the lines be black. The detailed real-time image is shown below.

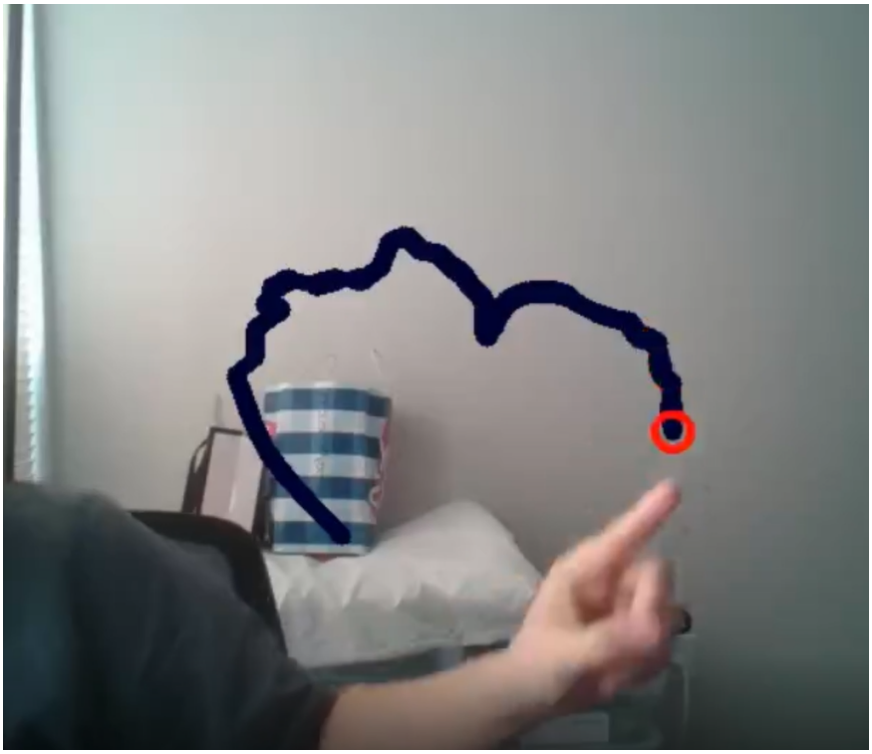

Fig Above: black lines and red bounding circle,

captured in a random frame of my demonstrate video

2. index and thumb all up – gun

In this pointer gesture, Same principle applies. If the most area is covered by ‘gun.jpg’, then the hand can be decided to be a gesture ‘gun ’and lead into the ‘gun’ function. There the color of the bounding box will be black while the lines will be red. The detailed real-time image is shown below.

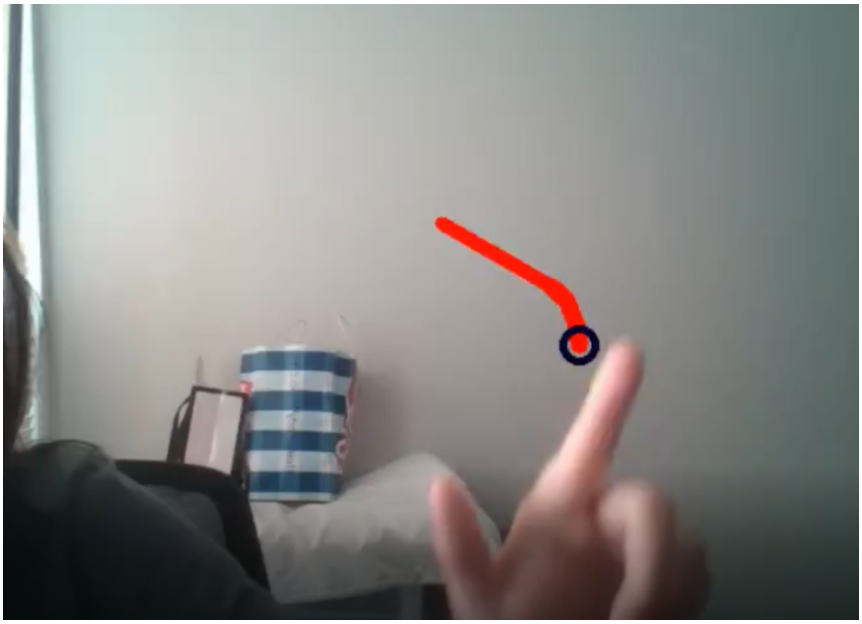

Fig Above: red lines and black bounding circle,

captured in a random frame of my demonstrate video

3. Index fingertip stays for 2s – lines erase.

   If the index fingertip stays for more than 2s, then the frame will be determined to be the third situation. All lines will disappear while only the red bounding box circle around the finger. The detailed real-time image is shown below.

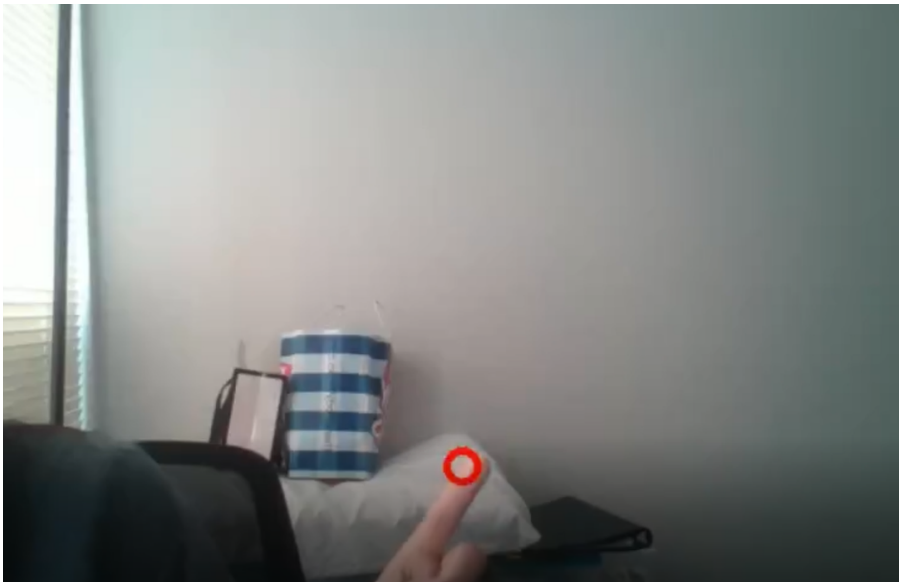

Fig Above: red bounding circle and no lines,

captured in a random frame of my demonstrate video

Connects the points from each frame and turn it into a smooth line

In this step, I first set up some hand gestures as I've mentioned before and connect the points of each frame. It can be easily done by collecting the fingertip position of the index finger of each frame. A list should be created to store all the information.

Here I also collect the distance of fingertip's movement from each frame and add them all together to get the total length of index fingertip movement. This step is to make sure my tracking line disappears when the length of the line is too long. To do so, I use the Pythagorean Theorem and apply it to every two frames.

The lines are set to different colors due to different gesture fits. Once the gesture is chosen, a kernel is designed to do the next image analysis step.

Turn off the video and all windows

This is a simple but necessary step. To successfully turn off the program, press ‘p’ and a keyboard interaction function is designed to achieve. I double destroy the window and release the camera after the keyboard function to make sure the computer camera is turned off while no window is left on the screen.

RESULTS

The result turned out to be a success and basically my program can follow all the goals.

Here’s some screenshot from my demonstration video.

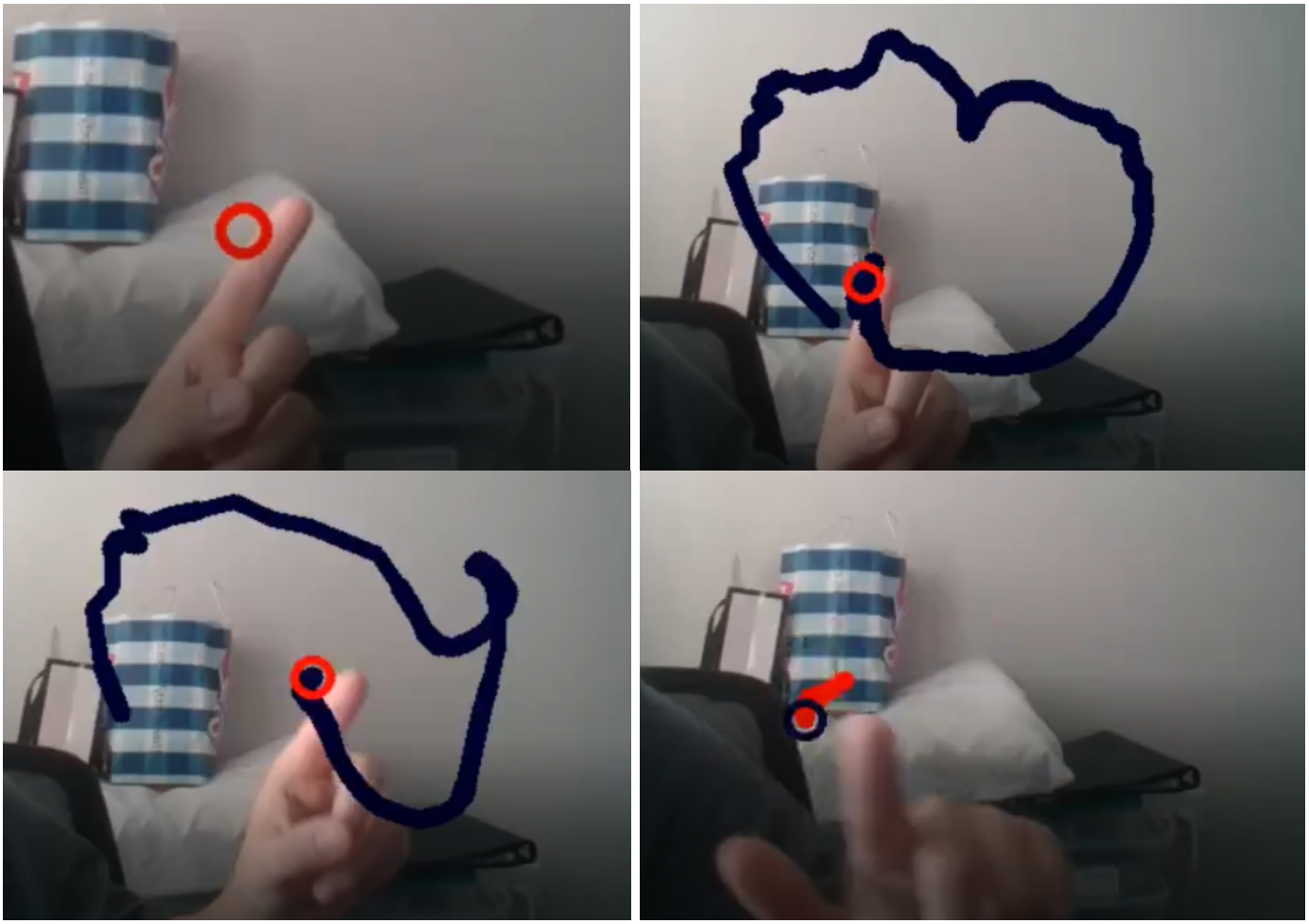

Fig Above: examples from my demonstrate video

FUTURE IMPROVEMENT

More gestures recognition

It would be better if we add more gestures and assign more functions on it,it won't take a long time but can significantly add more fun to this project.

For example, we can set a function which can detect if the hand is turning the 'pointer' gesture to left direction or right direction, and turn the current slides to the previous page or the next page accordingly.

A better way to find the index fingertips

Right now we make use of MediaPipe API, although we can also use the 'find the contour' strategy like dilation/erosion as taught in lecture, they are both not the best option. More effort should be done to refine the strategy of finding contour and isolate the hand. For now, one problem exists is that the program cannot clearly distinguish between our face/arm and hand because their color (rgb) are basically the same. Advanced methodology should be used instead of detecting hands by rgb.

REFERENCE

[1] Tran, D.-S., Ho, N.-H., Yang, H.-J., Kim, S.-H., & Lee, G. S. (2020, November 23). *Real-time virtual mouse system using RGB-D images and fingertip detection - multimedia tools and applications*. SpringerLink. Retrieved December 7, 2022, from https://link.springer.com/article/10.1007/s11042-020-10156-5

PERSONAL VIEW

ECE332 provides me with an introductory-level understanding of the fundamentals and applications of image analysis techniques.

From the very beginning, we learned about binary image processing, which lays a solid foundation for image analysis learning. And then we learned about color segmentation and region segmentation. We first deal with isolated images of black and white and then advanced real hand images. I think it's a good idea to start with simple images such as palm and gun in the MP2, that's easier for some new-learner such as me to get a step in.

Edge, contour, Hough transform and texture part is kind of hard for me, since I just get confused with all kernel things. It's not easy to extract the right information and build the correct kernel when I'm supposed to use the kernel to do a transform. I prefer there's a way to concretize kernel and transform, which would help me to better understand.

Motion and tracking is also a challenging part of this class. It takes a long time for me to translate video-to-image and image-to-video, but once it's done, these functions can be applied to any similar part.

In the class, professor Ying always tries to prove every algorithm he's mentioned, that's really helpful for me. Although most of the time I just follow his steps, his thinking method can be learned through class. And I like Professor Ying's idea of not always relying on Machine learning and packages. Reading and writing source code is the best way to understand it.

This course is helpful to my own research. Currently I'm doing research related to the mitigation of adversarial attack on Vehicle trajectory prediction. And it contains image and computer vision topics such as how to extract the information from the vehicle camera. For example, when we are trying to input the map data and vehicle road data. Sometimes we need to

make sure if the data is easily collected, or whether more detailed data can be extracted from the vehicle camera and road map. With the help of this course, I have a basic understanding in training images and video. I could determine which information can be easily obtained while some data is hard to collect.

If more time is given, I also want to explore fancy projects of computer vision in class. Currently we are mostly dealing with basic principles in CV. It would be cool if we can analyze a face recognition system or object detect project in lecture.

FEEDBACK OF THE COURSE

The course setting is very mature and contains a bunch of important knowledge if I want to further explore image analysis. And the MP is kind of a challenge, especially the last three. One thing which might make me anxious is the arrangement of homework is somewhat unevenly distributed. For the first few weeks, sometimes we have 2 MP for each week while the deadline of these two MP only gaps for two days. Although these MP's are easier in workload and difficulty compared with the other MP, it makes me a little anxious. Because if any problem occurs, there’s only two days for me to figure it out. I think it might be better if each MP has more than three days extended time.