CS4187 Computer Vision and Interactivity

Homework Assignment #2

WANG Yinuo 55670000

**INTRODUCTION**

**After learning about computer vision for a period of time, this assignment is to ask us to use the OpenCv Framework to carry out basic edge detection method on pictures, videos and capture image from a webcam.**

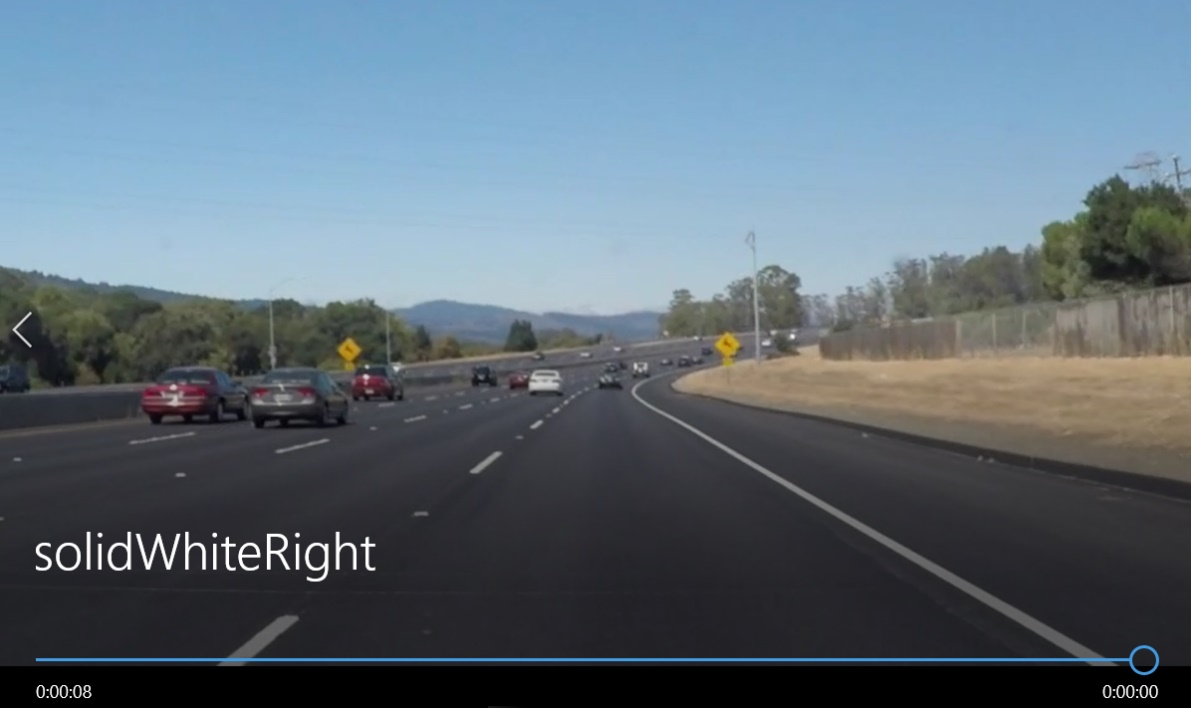
**The first task is to** process the video “solidWhiteRight.mp4” and detect the lane lines on the road.

Figure Capture of solidWhiteRight.mp4

The second task is to capture the real-time color image stream from a webcam. In addition, the program should detect the circles in the paper model of the glasses, and draw the following pattern at the position of the circle.

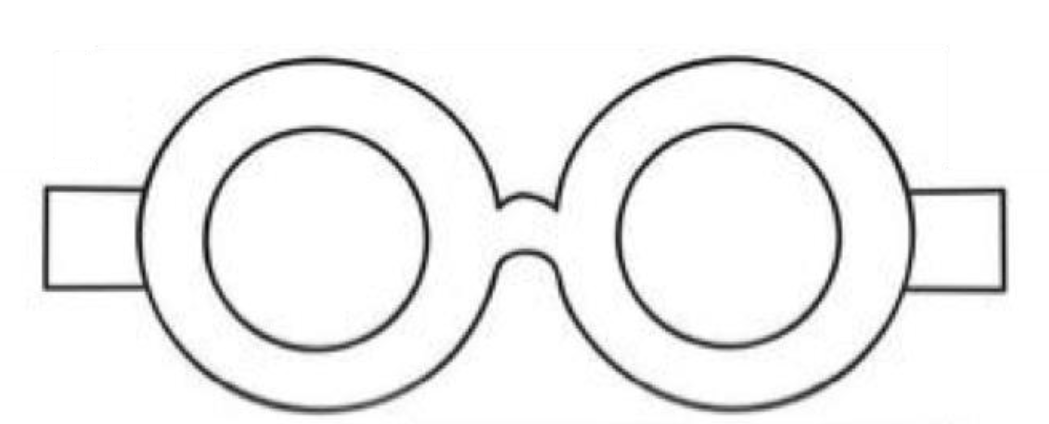


Figure 3 Eye

Figure 2 glasses

The last task is to process the photo that a circle will be moving smoothly along the line sketched in the photo.

**RELATED WORK**

1. OpenCV Framework and c++ learning

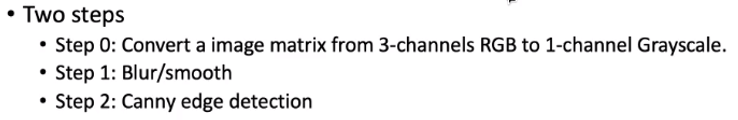
2. Basic knowledge of edge detection

3. Basic knowledge of line/circle detection

**APPROACH**

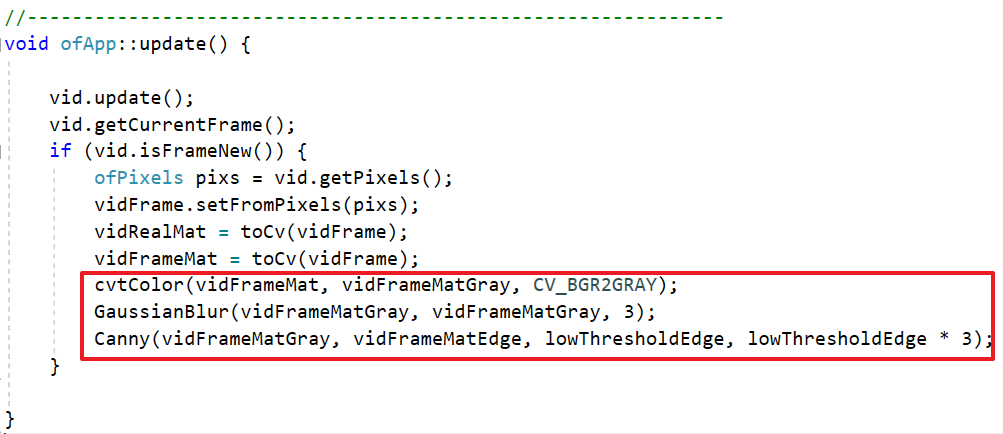
1. Detect the lane lines on the road

There are three steps to detect edges of one picture:

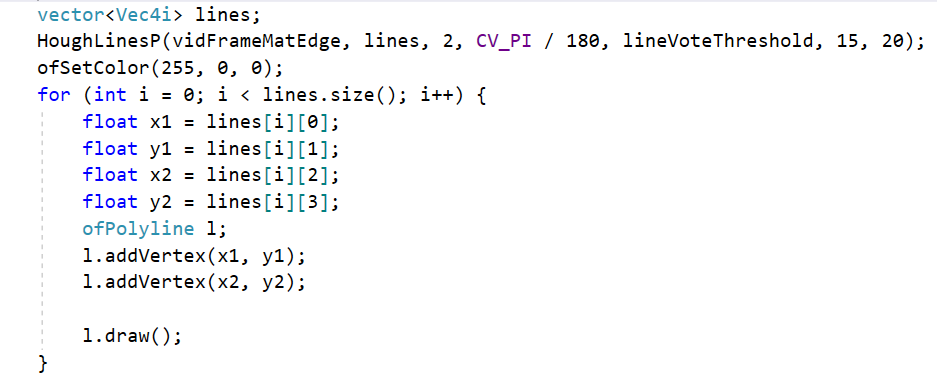


So does video.

The main code is as follows, where vidFrame is one frame of the video.



To detect the lane lines, we use Hough Transform, which can identify straight lines in an image. The code is like this:



The detection result is shown below.



In addition, I added two sliders to adjust the two thresholds, one for edge detection (low threshold) and one for line detection (line vote threshold).

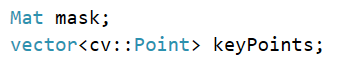
As these figures shown, when we increase the value of low threshold, the edges which have been detected decrease. This is because the low threshold will filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value.

Besides, when the line vote threshold decreases, the number of lines detected increases. The reason is Hough transform finds out lines through the voting mechanism. When the line vote threshold decreases, the noise will increase, so the number of lines will increase.

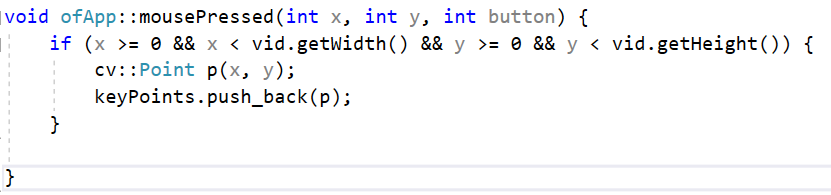
The program also allows that click the mouse to interact with the video to display edge/line detection of the area of interest, as shown below.

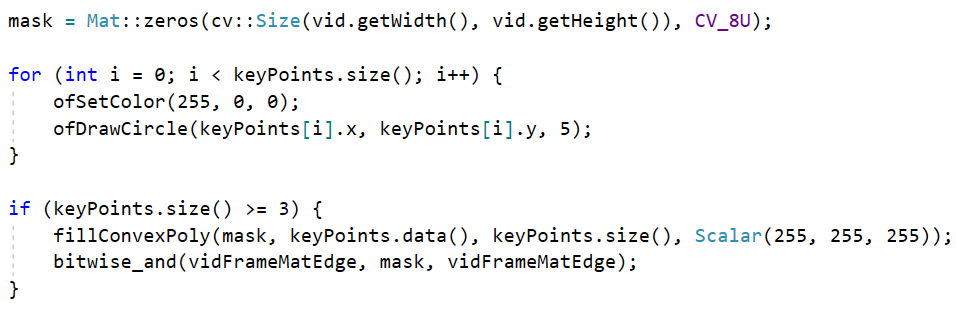


To accomplish this task, we first define a mask and a set of keyPoints.



The basic logic is that when the mouse clicks on a point in the video, it forms one keyPoint, and when the number of keyPoints can form a region that we are interested in, the program will create a mask.

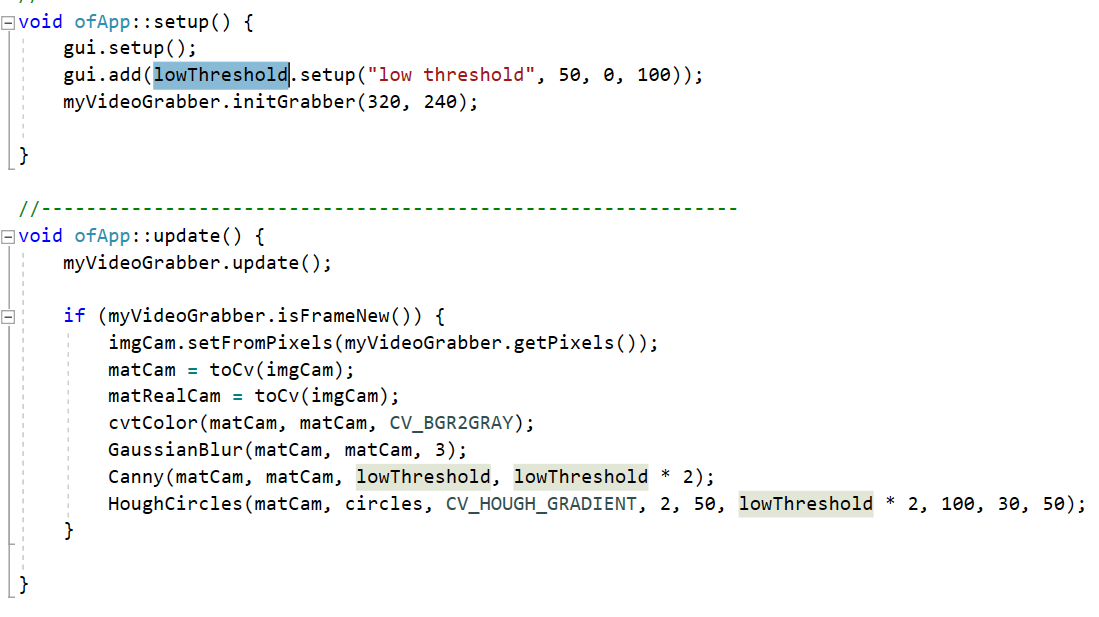




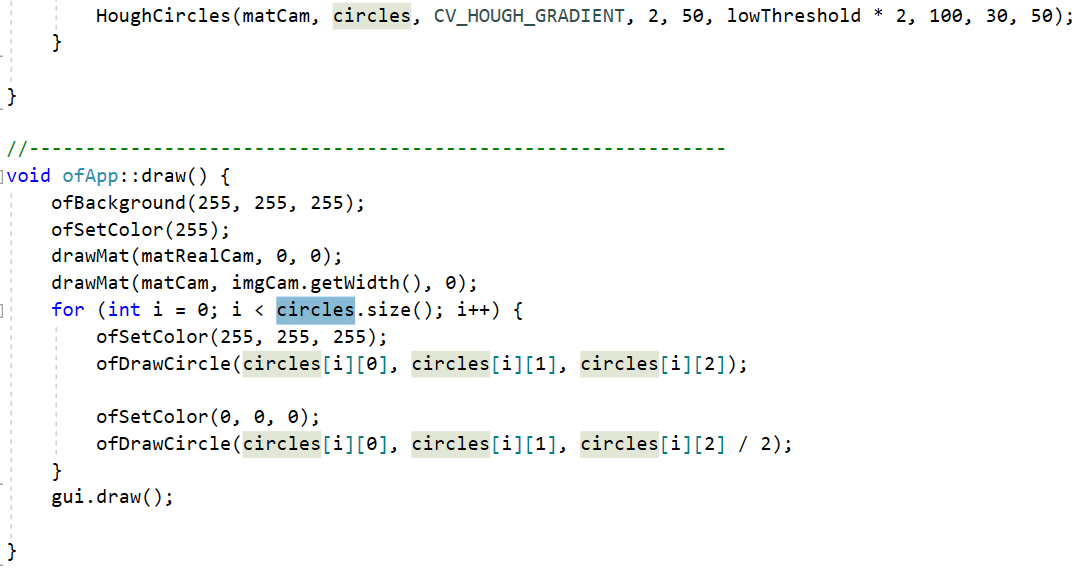
In this area that we are interested in, lanes will be marked with red lines.

In task 2, we should capture the color image stream from the Webcam in real time and do the edge detection where the threshold can be changed with a slider.

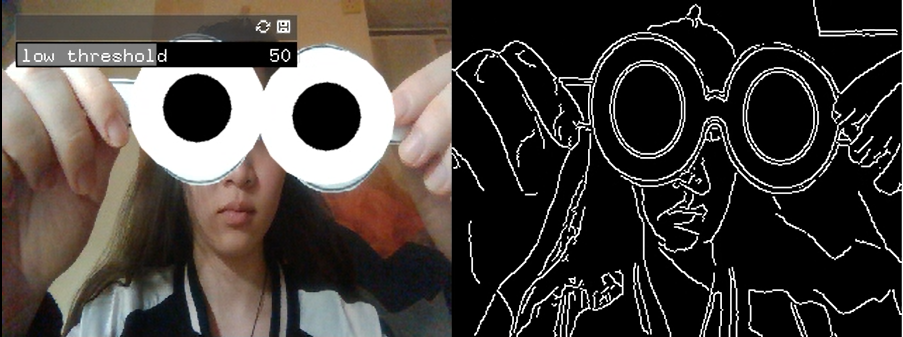
Here is the main code.



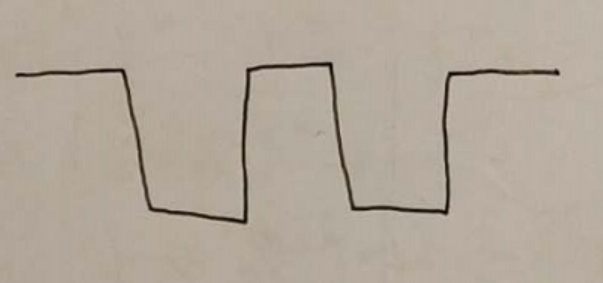
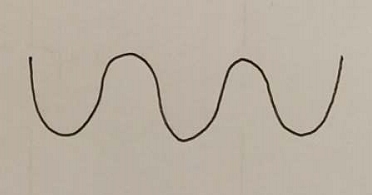
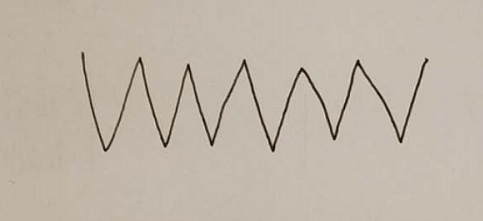
In addition, we need to detect the circles (a pair of round glasses) in the image and add "eyes" on the glasses.



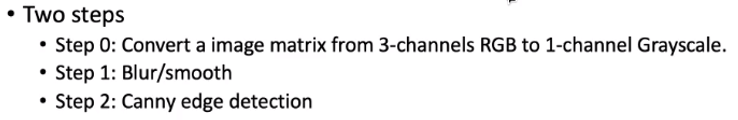
The running results are as follows:



To complete task 3, we should process a photo which contains a line sketch on a piece of paper. My sketches is shown below (test1, test2, test3).



Firstly, we should use the three basic steps to detect edges and also add low threshold sliders to filter out the edge with highest gradient value.



In order to adjust the threshold at all times to make edge detection go more smoothly, I put the detection steps in update(). If the threshold changes, the program will do edge detection again and then updates the endpoints of the line in picture.

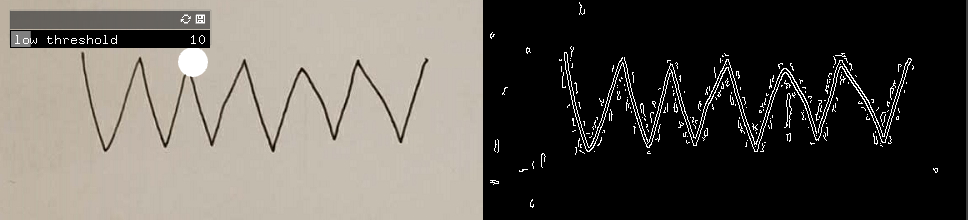


Figure 1 detect with lower threshold

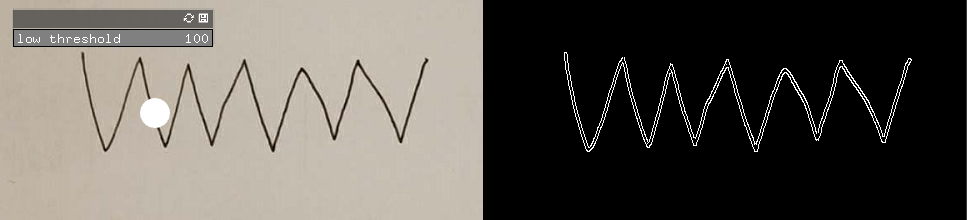
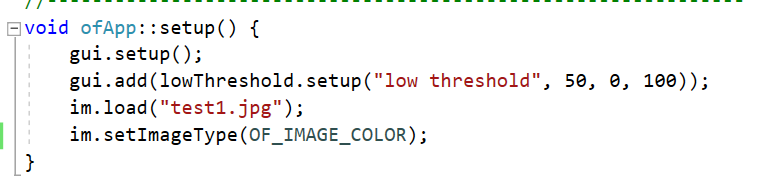
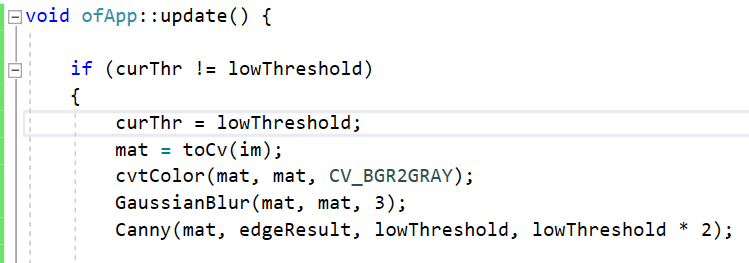
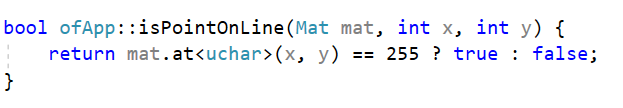


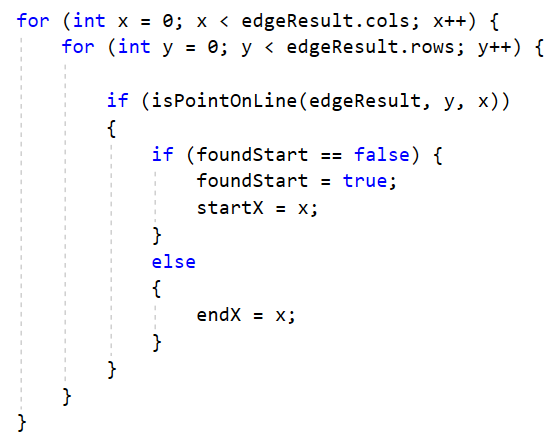
Figure detect with higher threshold

The main code is as follows:

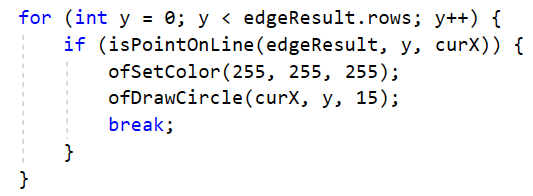
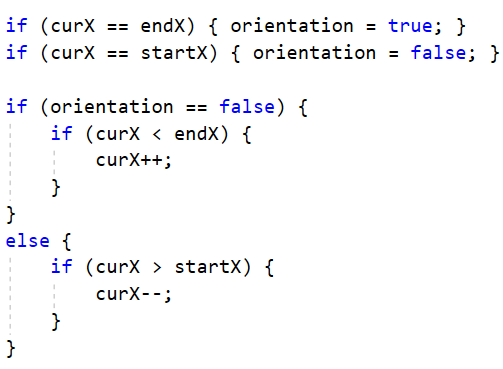




And then I write a loop to go though all the pixels which has been detected as a point on the edge (use function “isPointOnLine” to tell) to find out the X-axis values of start point and the end point.



Finally, I go though the picture again with a small circle and update the circle’s position (curX and y) time to time which is within the startX and endX to see the trajectory of the egde.

Also, in order for the small circle to go back after touching the endPoint, I created a Boolean variable “orientation” to check whether the small circle should turn the direction.

To adjust the threshold, the circle will go though the edge smoothly.

