TASK 3

In this code, we have integrated colour detection with path following such that it has the same sense of direction as it did and along with it the code also constantly checks for the visibility of the barricade colour, seeing which it will print “Stop”. When other colours are shown and no path is visible, the code prints “Go straight”, since doing so will allow the bot to come back to the clear track.

We needed to ensure that the bot doesn’t stop if there is some noise due to which the program detects small amounts of the barricade colour, since that would be a hinderance to the motion. So, we calculated the area of the books we were using while testing using the contourArea function, and we provided an if statement that only allows the bot to print “stop” if the detected colour contour is larger than a certain area. There could be a need to change this value based upon the environment we will be using the bot in.

We also needed to make sure that the bot prints “Go straight” when it cannot detect the path since it is being blocked by an unwanted colour, since moving straight would allow the bot to come back to the clear track and continue proper motion. Hence, we added many if-elif-else statements so that this is taken care of.

One issue we faced for a long time while running this code was that “Stop” was being displayed along with the other direction cues the code was getting. We used a break statement to come out of the for loop, but the code was running in an outer while loop and thus it kept entering the for loop again. Not using the for loop at all wasn’t leading us to the desired output either, and we had a backup plan to allow the code to quit or run and infinite while loop when it senses the stopping colour, but fortunately we didn’t need to do that since we tried another approach in which we defined a variable and changed its value when it entered the block that prints “Stop”. This change of value of the variable restricted the code from entering the for loop again and now it runs smoothly.

We defined a function that returns the masked image of the colour it gets as its input, and this masked image is used later in the code for making contours. The input to this function is given by the value received from the google sheets, and this way the code works perfectly as desired. The lower and upper hsv limits for each of the colours that we used were found using trackbars and are only for the three notebooks that we used while testing. For different situations where other colours may be used, we may need to change the lower and upper hsv limits.

In the folder for task 3, along with the main python files that read video files and use webcam, we have also included the JSON file and the input video that was used in the code for testing. We have put the completed video which was our test, but the input video is also provided so that anyone using the folder can test out the code without much hassle.