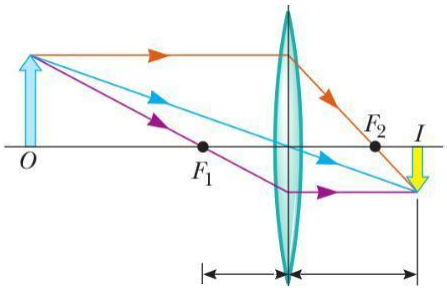
Report 2 - Nihal Afsal

Lane Line Detection Using a Camera

1. A converging lens has a focal length of 10.0 cm. An object is placed 30.0 cm from the lens. Find the distance of the image from the lens and the magnifcation of the image. (*15 pts)*



Focal length = 10 cm

Object Distance from lens = 30 cm

$$\frac{1}{v} - \frac{1}{u} = \frac{1}{f}$$

$$\frac{1}{v} = \frac{1}{f} + \frac{1}{u}$$

$$\frac{1}{v} = \frac{1}{10} - \frac{1}{30}$$

$$\frac{1}{v} = \frac{2}{30}$$

$$\frac{v}{1} = \frac{30}{2}$$

$$v = 15$$

$u = 30$

magnification $= \frac{-v}{u}$

magnification (M) $= \frac{-15}{30}$

- Converging lens magnification is always negative, hence magnification is always seen as negative.

2. Find a camera and lens suitable for an autonomous vehicle. In 2-3 sentences, describe its properties and why you believe it will work well. *(10 pts)*

- The Basler Ace 2 Pro is an example of a camera and lens appropriate for an autonomous vehicle. Unique features of Basler Ace 2 Pro include Compression Beyond and Pixel Beyond. The camera can use bandwidth more efficiently, thanks to Compression Beyond. Videos can be taken at greater fps if the camera's internal bandwidth is more effective. Better motion tracking and object detection are made possible by cameras with higher frame rates, enhancing perception and decision-making for autonomous vehicles. Additionally, Pixel Beyond allows users to alter pixel size and resolution, enabling autonomous vehicles to increase the accuracy and quality of sensor data and hence increase their capacity to detect and identify items in their environment.
-

3. Create an image processing pipeline in Google Collab that uses 3+ OpenCV functions to manipulate the image and identify the lane lines. Use one of the images provided on Elearning. *(30 pts)*

-

As you can see, by identifying the center lane lines in the image, my existing pipeline performs well. I am still attempting to find out how to take the outer lane lines, but I have some trouble doing so. The code detects the lane lines by utilizing a single function named process image and a number of OpenCV conversions and manipulations.

4. Apply your image processing pipeline developed in problem 3 using VS Code to one of the video files provided on Elearning. Refine your code until you are happy with the qualitative lane line detection accuracy. Turn in an mp4 file of the result as well as a discussion of how it works and how it performs. *(30 pts)*

- Consequently, the pipeline I used for my image showed promise but entirely failed when I used it for the video. I tried to improve it by concentrating on the white and yellow lanes, but my program had trouble understanding the white lines when the video was filmed in low light. I made every effort to get it to recognize the lines, but in the end, I gave up because I was irritated.

Original Pipeline Video:

https://drive.google.com/file/d/1eIrpXW9yzY0DrWGJ8jfPdEpM52VzqO43/view?usp=sharing

Modified Pipeline Video:

https://drive.google.com/file/d/1nhPtVsttLLPDFbI3lVsQcbl48OAcD7pV/view?usp=sharing