Assignment 1: Report
Computer Vision COL780

1. **Background Subtraction** (background_subtraction.py)

In this python program, Background subtraction for the given videos has been accomplished. The OpenCV library function “createBackgroundSubtractorMOG2” has been used.

- *DetectShadows:* parameter is set to True and shadows from the frames are removed by thresholding and removing the grey pixels from the black and white image.
- *History*: This parameter includes 10sec of the initial video. So, since fps is 30 frames/sec, history is 300 frames.
- *VarThreshold*: This is set to 22 whereas the default is 16.

Further Median filtering was also tried to remove much of the speckle noise.

2. **Morphological Cleaning** (image_cleaning.py)

- Techniques like **erosion, dilation, opening, closing, gradient, top-hat and black-hat** were tested
- Finally ‘**opening**’ with a kernel size of 3, was chosen as the technique to accomplish image processing.

3. **Edge Detection** (edge_detection.py)

Edge detection is achieved by applying Canny edge detection. **Laplacian, Sobel-x, Sobel-y** operators are also tried upon and tested for best results. The main difficulty lied in chipping the threshold values used in **Hysterisis** step of the edge detection. Since higher the average intensity of the image, more are the white pixels. Hence a higher threshold should be used. So we decide to dynamically choose the thresholds.

- *Lower threshold*: max(0, (1.0 - sigma) * v) where v is mean intensity of the image
- *Higher threshold*: min(255, (1.0 + sigma) * v) where v is mean intensity of the image.

4. **Line Detection** (hough.py)

Line detection applied on the mage contained from the last step, is accomplished using the Probabilistic Hough Transform. Simple hough transform was also applied, but results were observed to be better in the probabilistic one.
After this file runs, we receive a set of frames where each frame is marked with the bundle of straight lines detected by the hough transform.

5. **Medial Axis Detection** (hough.py)

In the same file named ‘hough.py’, we added the code for medial axis detection. The parameters for HoughLinesP (Probabilistic hough transform) for each file are chosen as follows:

- *Threshold*: 60
- *MinLineLength*: max(avg_length,100) where avg_length is a factor * the average length of lines found from the previous iteration satisfying the conditions specified.
- *MaxLineGap*: 100

Voting Mechanism for angle constraint:

- After we get the lines satisfying this condition, we find the subset of lines which are nearly parallel and have the highest frequency.
- This has been done by forming a new voting system with lines entering bin ‘i’ if ‘i’ = slope of line in degrees/10.
- These lines will be along the rod in the video and hence should be chosen and the others must be rejected.

Choosing the two extreme edges of the rod:

- Our of the lines obtained from the above step, we choose the ones with lowest and highest y-intercept. This pair of lines corresponds to the two extremes of the rod.
- Next, we chose the medial axis as the average of these two extremes.

Smoothing over frames:

- Since the medial axes have been found independently for every frame so there will be lot of noise in its position
- We used averaging with a window size of 5 (each side) over the frames to get a stable orientation.

Link to the best 4 videos: click here

Or

https://drive.google.com/drive/folders/1CdwN62krAx2ye18hTxSmNDiJl3pdfJOB?usp=sharing

Sharut Gupta (2017MT60250)
Harkirat Singh Dhanoa (2017CS50408)