Constantine Kazakov - 321827834 | Yosef Golubchik - 209195353 | Denis Shapira - 205356801

Lane Detection Project

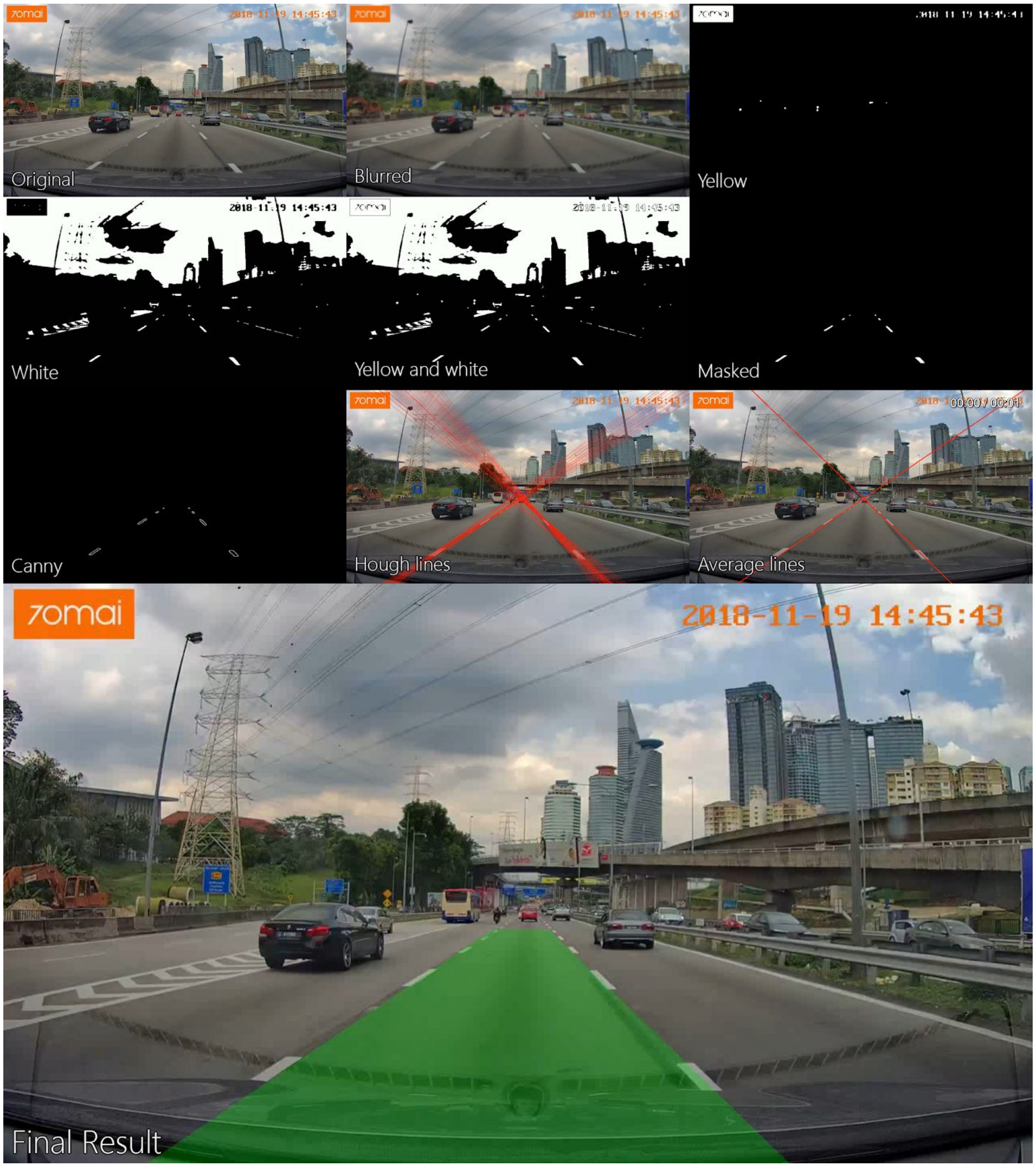
https://github.com/JosephGolubchik/CV-Lane-Detection-Project

Introduction

In this project we attempted to create a program that given a video shot from a car, detects in each frame where the lane boundaries are. Detecting lane lines is extremely useful and necessary for driverless or autonomous cars, which can for example use this information to know how well the car is staying inside its lane and if it needs to turn and how much.

This problem has been tackled using classical computer vision algorithms, but the best solutions were created using deep learning. We decided to do our best using classical computer vision algorithms.

We tried two methods:
1. Using canny edge detection and hough transform to detect lines.
2. Using homography to obtain a bird’s eye view of the road and detect the lines using sliding window algorithm.



https://github.com/JosephGolubchik/CV-Lane-Detection-Project/blob/master/lane_Detection_mask_workflow.png?raw=true

Approach and Method

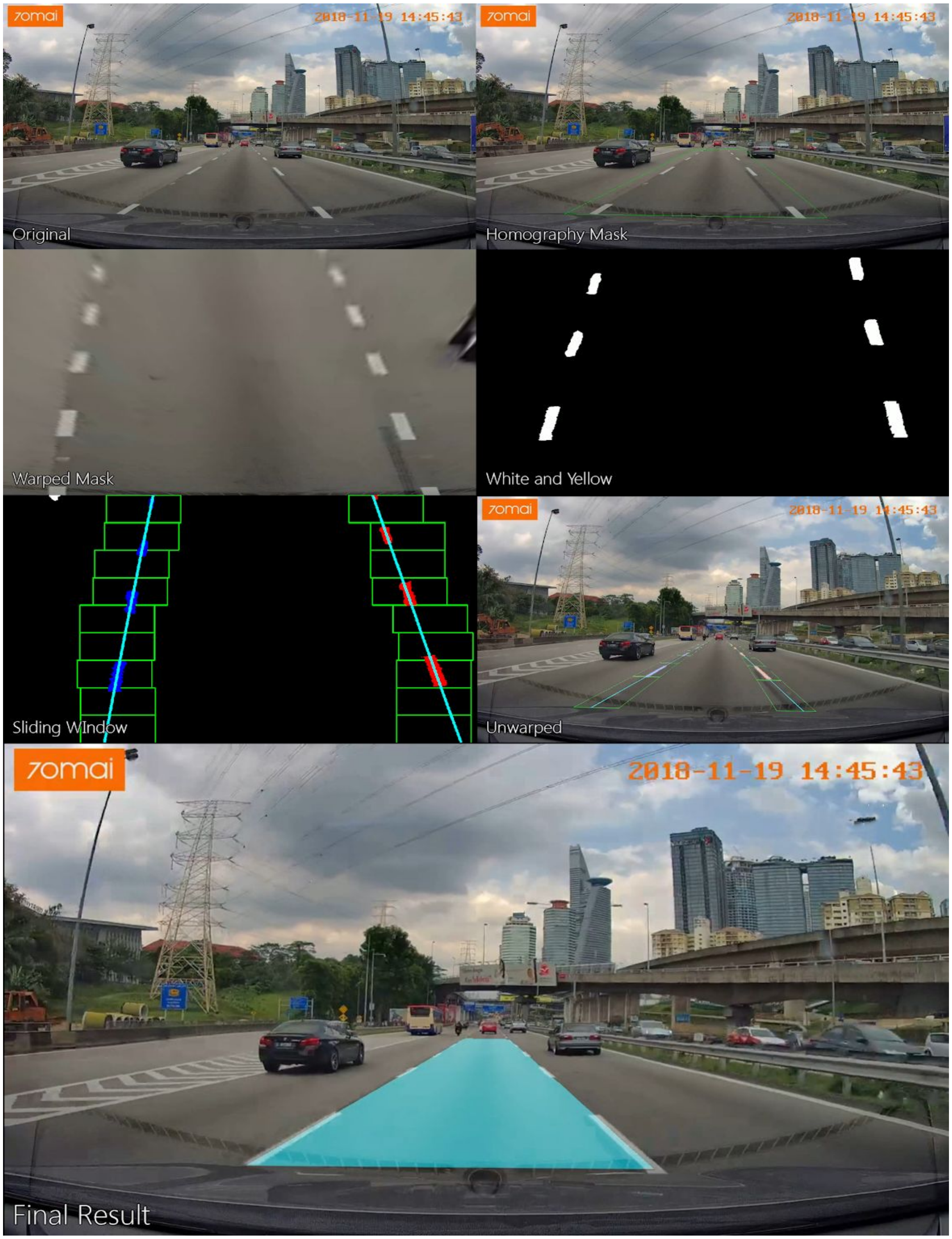
In both methods we first need to calculate the distortion of our camera using a picture of a chess board taken from that same camera, and undistort each frame before applying the algorithm. We weren't able to film our own videos for the project, so we could undistort the frames. But the code for detecting and removing distortion can be found on our GitHub separately.

First Method - Canny Edge Detection and Hough Transform

This method is very simple to implement and understand, but it's very limited:
1. It can't properly detect curved lines - only straight lines, so if the car is turning the lane detection becomes unreliable.
2. The code needs to be modified and changed for each individual video, because it uses a handmade mask image that needs to be made for each video, and sometimes there isn't one mask that works throughout the whole video.

Algorithm:
1. First gaussian blurring is applied, to help edge detection later on.
2. Lane lines can be either white or yellow, so we need to extract only white and yellow pixels. We get the yellow pixels by converting the image into LAB color space, and leaving only pixels where $B > 160$.
3. White pixels are extracted by converting the image into HSV color space, and leaving only pixels where saturation is fairly low and value is fairly high.
4. Canny edge detection is applied.
5. Lines are detected in the edge image using hough transform, and lines that are almost completely vertical or almost completely horizontal are discarded, because lane lines are practically never like that.
6. Lines are divided into groups where two lines are in the same group if the distance between them is smaller than a given threshold.
7. An average line is calculated for each group.
8. If we get more than two groups, we choose the best two by taking the two which are closest to the middle of the lane, which are usually the desired lines.

9. The final two lines are calculated by combining the current best two with the previous frame’s best two, to get a smooth transition between frames, and to prevent lines flying all over the place if there was a mistake in the detection in one single frame.
10. The intersection point of the best two lines is calculated and the area between the lines is colored, and we get the colored lane.



https://github.com/JosephGolubchik/CV-Lane-Detection-Project/blob/master/lane_detection_homography_workflow.png?raw=true

Second Method - Homography and Sliding Window

This method is a little harder to implement than the previous one, but also works better, though is also somewhat limited:
Each video requires a different selection of points for the homography, which is selected manually and not automatically, which also doesn't always work throughout the whole video.

Algorithm:
1. Four points are selected to be transformed to bird's eye view using homography.
2. Homography is applied to obtain a bird's eye view of the road to more easily detect the lane lines.
2. Lane lines can be either white or yellow, so we need to extract only white and yellow pixels. We get the yellow pixels by converting the image into LAB color space, and leaving only pixels where $B > 160$.
3. White pixels are extracted by converting the image into HSV color space, and leaving only pixels where saturation is fairly low and value is fairly high.
4. We apply the sliding window algorithm which obtains for each lane line some points along the line, and a fitting polynomial equation is calculated for each line using np.polyfit().
5. The area between the two lane lines is drawn on the bird's eye view image of the road.
6. The image is transformed back into the original perspective using the inverse of the homography matrix from the earlier homography, and is placed on top of the original image.

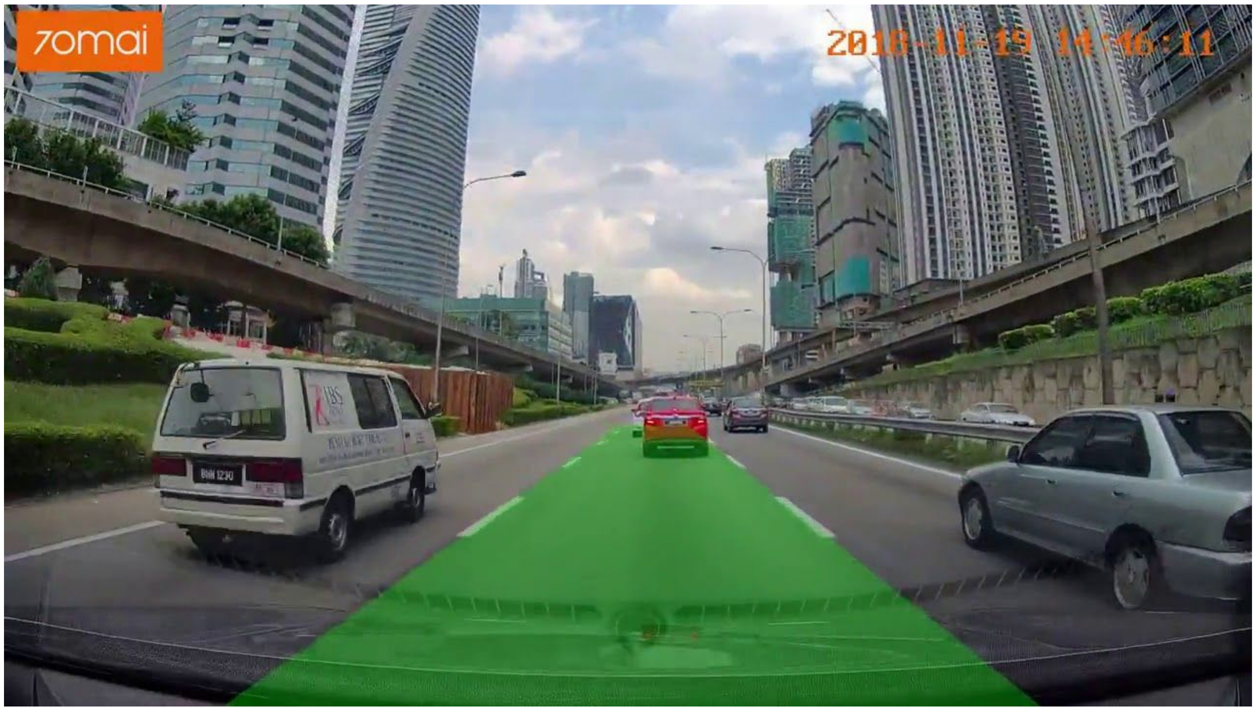
Limitations of both methods

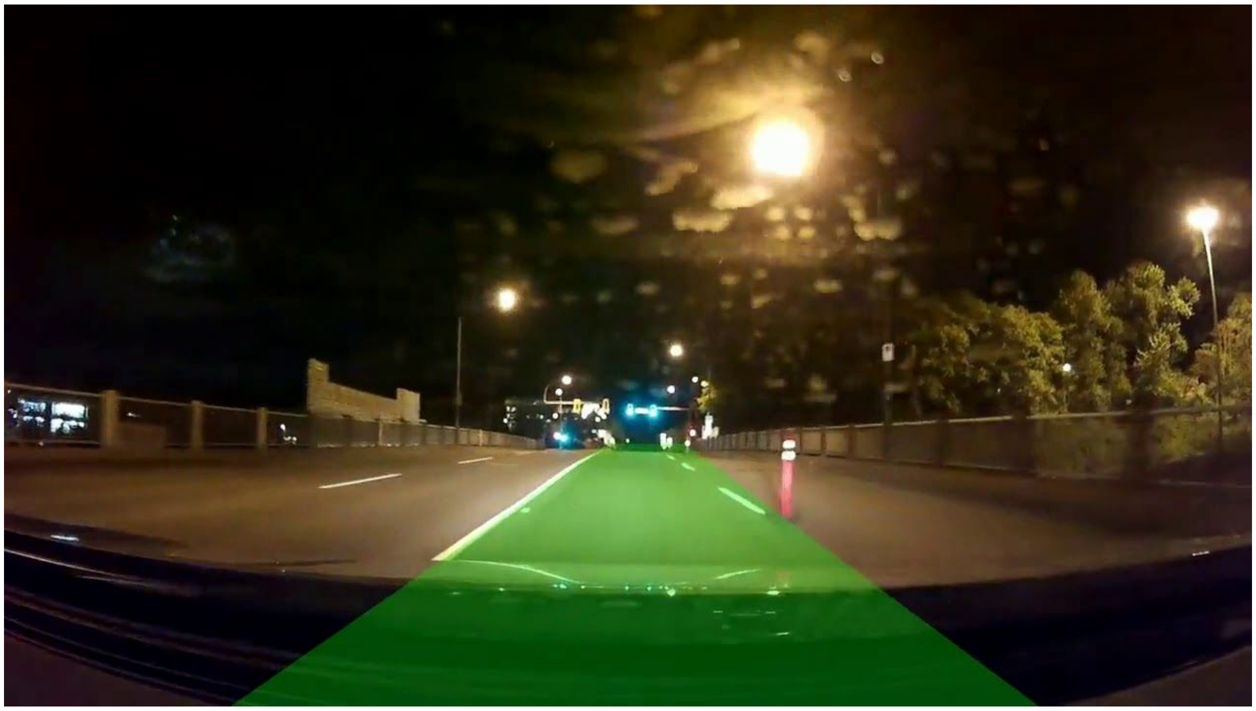
1. Doesn’t work correctly most times when the lighting changes suddenly (entering a cave, going under a bridge…). This is because the parameters of canny edge detection, hough, and white and yellow pixel thresholding need to be tuned for each video, and when the lighting changes drastically during the video the parameters aren’t correct anymore.
2. Can’t properly detect lines when there are symbols or text on the lane, which also create strong lines and to the algorithm look just like lines that belong to the lane lines.misses lines, o
3. Both methods depend on a manually selected mask in order to work, but many times the mask only works on part of the video, and on other parts doesn’t cover the whole needed area and r covers too much and detects unnecessary lines.
4. Going close to other cars leads to the mask being partially or almost fully covered and then the algorithm can’t detect the needed lines or even detects lines that belong to the car in front.

Results

First Method - Canny Edge Detection and Hough Transform

(Click on image to go to video on youtube)





Results

Second Method - Homography and Sliding Window

(Click on image to go to video on youtube)