EE 417 COMPUTER VISION

TERM PROJECT REPORT

EFE ÖZTABAN – 25202

KAYRA BİLGİN - 25117

1. Importance of the Problem and Problem Definition

a. Importance of the Problem

One of the most significant problems of the 21st century is traffic. The crowded population in the metropolises with the increase in urbanization causes traffic. This high density in the traffic makes it very fragile. Small mistakes and violation of the traffic rules can cause really crucial problems. Thousands of people die every day due to traffic accidents which are caused by the violation of the traffic rules. That's why traffic controls and traffic fines are very essential in order to prevent these violations. However, it is very hard and almost impossible to check traffic all the time by officers. Technologies are being developed as solutions to this problem, but they have not been completely successful yet.

The most optimal and permanent solution to the traffic violation problem is to raise awareness of all people about obeying traffic rules. However, this would be an overly optimistic approach. For this reason, it is a more logical approach to detect drivers who violate traffic rules and create deterrent penalties. Such constructive punishment systems are implemented all over the world, but these traffic rule violation controls are not a very deterrent since they are not effective at all. In addition, doing these controls by human hand is not possible in terms of inadequate human power and challenging working conditions. For this reason, the control mechanism that needs to be developed should be automated and work 7/24. In addition to that, all driver behaviours that cause traffic violations should be detected such as speed limits, wrong way driving or breakdown lane violation.

Computer Vision is a very important tool for traffic violation detection systems. Traffic cameras and image processing & computer vision algorithms are widely used in traffic control and violation detection systems and applications. Control systems to be developed using Computer Vision will eliminate the traffic problem in the near future. The use and development of such applications is highly supported by developed and developing countries. For this reason, although the population of metropolises increases day by day, the traffic problem will not increase at the same rate, on the contrary, it is expected to decrease and disappear with these technologies.

In our country, systems that monitor traffic rule violations such as EDS (electronic monitoring system) are used. However, this system used in our country needs a person or persons who will stand by the system and manually check the violations. Since the system, which aims to reduce traffic rule violations with Computer Vision techniques, does not need people, it becomes a more affordable automated system for the future.

b. Problem Definition

As it is explained in the previous section, image processing and computer vision techniques are being used in traffic violation detection systems. For this project detection of the traffic violations is chosen for the main problem. Some of the basic traffic violations are chosen for the project.

The project purposed to detect specifically the traffic violations listed below:

- Exceeded speed limit
- Breakdown lane violation
- Wrong way driving detection
- High occupancy of left lane in low speed

For detecting these traffic violations, a casual traffic road type is selected. System will work on the roads which have 3 lanes and 1 breakdown lane. To be able to test the system, a sample video of traffic on a 4-lane road (3 normal lanes and 1 breakdown lane) is selected. Sample footages from the video is given below:

Sample video is taken from the website given below:

https://www.youtube.com/watch?v=QuUxHIVUoaY&t=1571s

The initial problem is detecting the road structure in terms of boundaries and lanes of the road. Then, foreground detection and moving object detection problems will be studied in order to solve the general problem. After these, specific features should be extracted both from the road and the moving vehicles. For extracting these features computer vision algorithms will be used. With the extracted features, special conditions will be checked to detect the traffic violations listed above.

As a summary, in this project, it is aimed to create a basic traffic violation detection system with implementing computer vision algorithms to be used in traffic regulation applications.

2. Proposed Solution Methods

The main solution of the problem will be using image processing and computer vision algorithms to feature detection and extraction. Furthermore, using algorithms to analyse these features to detect traffic violations. In this section proposed solutions to the problem are explained generally and details about the solution methods are given as a list.

The system will use video frames to extract features and motion detection. In order to detect these violations, traffic lanes of the road will be detected with line detection. From the detected lanes, special ones will be determined such as breakdown lane or left lane in order to detect violations related to these special lanes.

With using a part of the frames from the sample video, the foreground of the road will be detected. With this foreground, moving objects will be searched from every frame of the video. When a moving object is found, special features about the moving object will be determined. Motion direction, speed, lane which they are moving on are some of these features.

Motion direction and speed of the objects in the traffic will be determined by checking their location in every frame. Detected location of the moving object will be compared with the lane boundaries and the lane they are traveling on will be determined. If a vehicle is moving on the breakdown lane, this violation will be reported using this information. Also, special checkpoint lines on every line will be used for speed detection. Time for traveling from one special checkpoint to another will be recorded and speed of the object will be calculated with this information. Found speed of the objects will be compared with speed limits of the road. In order to create a more regular traffic flow, a minimum speed limit will be set for the leftmost lane used for overtaking and fast passing, and a warning message is generated for drivers staying under this limit.

Also, motion direction of the object will be checked by the developed system and wrong way driving violations will be reported if the direction of the motion is not correct. These techniques will be combined to detect the traffic violations listed above and the system will report the detected violations.

1. Line Detection

- Line Detection will be used for detecting the road and lane boundaries. X and y coordinates of these boundaries will be recorded in the related variable for road calibration of the system.

- In addition to that, special checkpoints from every lane will be determined in this step. Real distance between these checkpoints and pixel locations of these checkpoints will also be recorded in special variables. These checkpoints will be used for speed and motion direction detection

2. Video Frame Processing Techniques

- Frames of the input video will be used for both creating the foreground of the road and detection of the moving objects and its special features.

- These frames will be processed by applying image processing tools to make them more efficient. In this way, frames will be more applicable to be used in computer vision tools.

- Smoothing methods such as Gaussian filtering and different filtering techniques will be used to process the frames.

3. Foreground Detector

- This system will examine the frames to find the foreplay of the road without any objects.

- With founded foreplay, moving objects on that foreplay will be detected more efficiently.

4. Moving object detection

- Using the found background, moving objects will be detected for each frame in the video. The positions of these objects in the image will be determined.

5. Detection of lane of moving objects

- The position of the found objects and the previously determined road data (lane locations) will be used to determine which lane they are in. In this way, if there is a vehicle using the safety lane, a warning can be created. Also speed of the cars will be determined for every lane.

6. Determining the distance and speed of objects

- Time will be calculated for vehicles passing between two zones determined for each lane on the road. Thus, the direction and speed of the vehicles can be determined.

- In this way, the situation of driving in the opposite direction, speed limit violation and being below the optimal speed in the left lane will be detected.

7. After violation detection

- Moving objects will be displayed during video processing. In addition, when any violation is detected, a warning will be given, and a photo will be taken to record the traffic violation with the related message.

3. Implementation Details

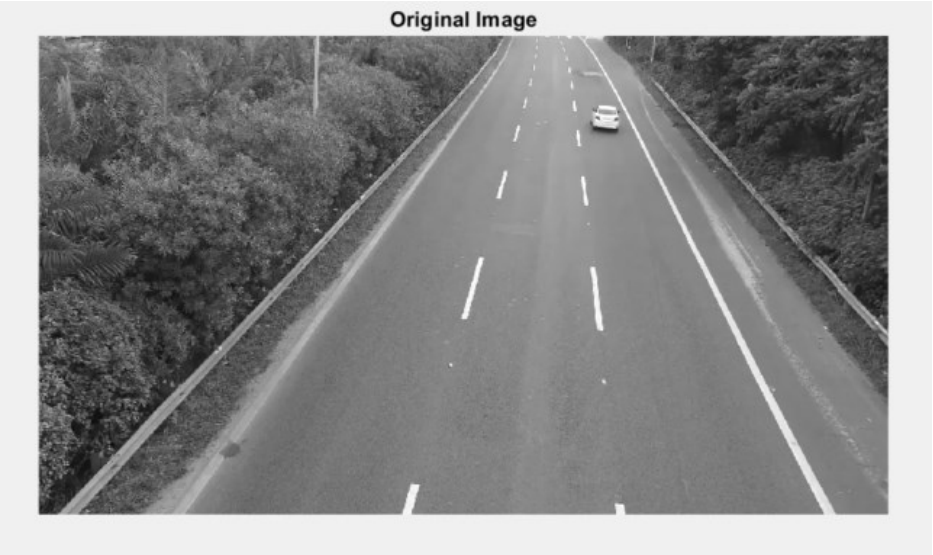
1. Line Detection

- As it is explained for road and lane characterization, line detection is used on the road image.

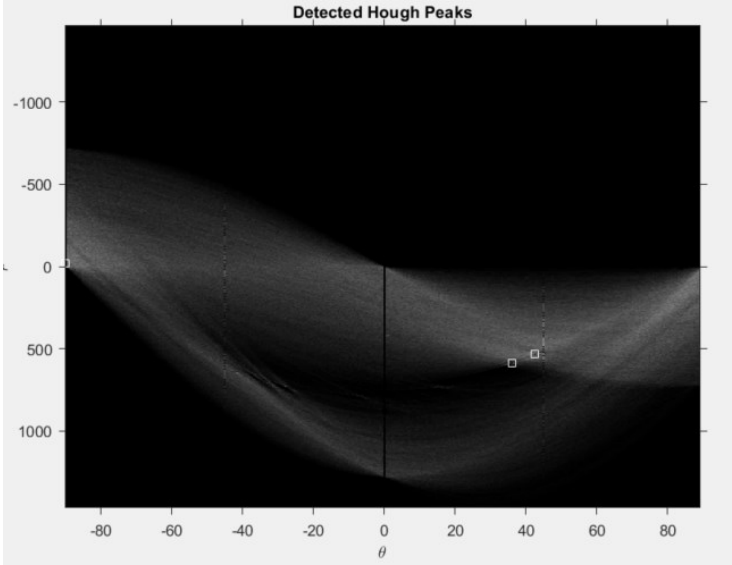
- As the line detection method HoughLines are used.

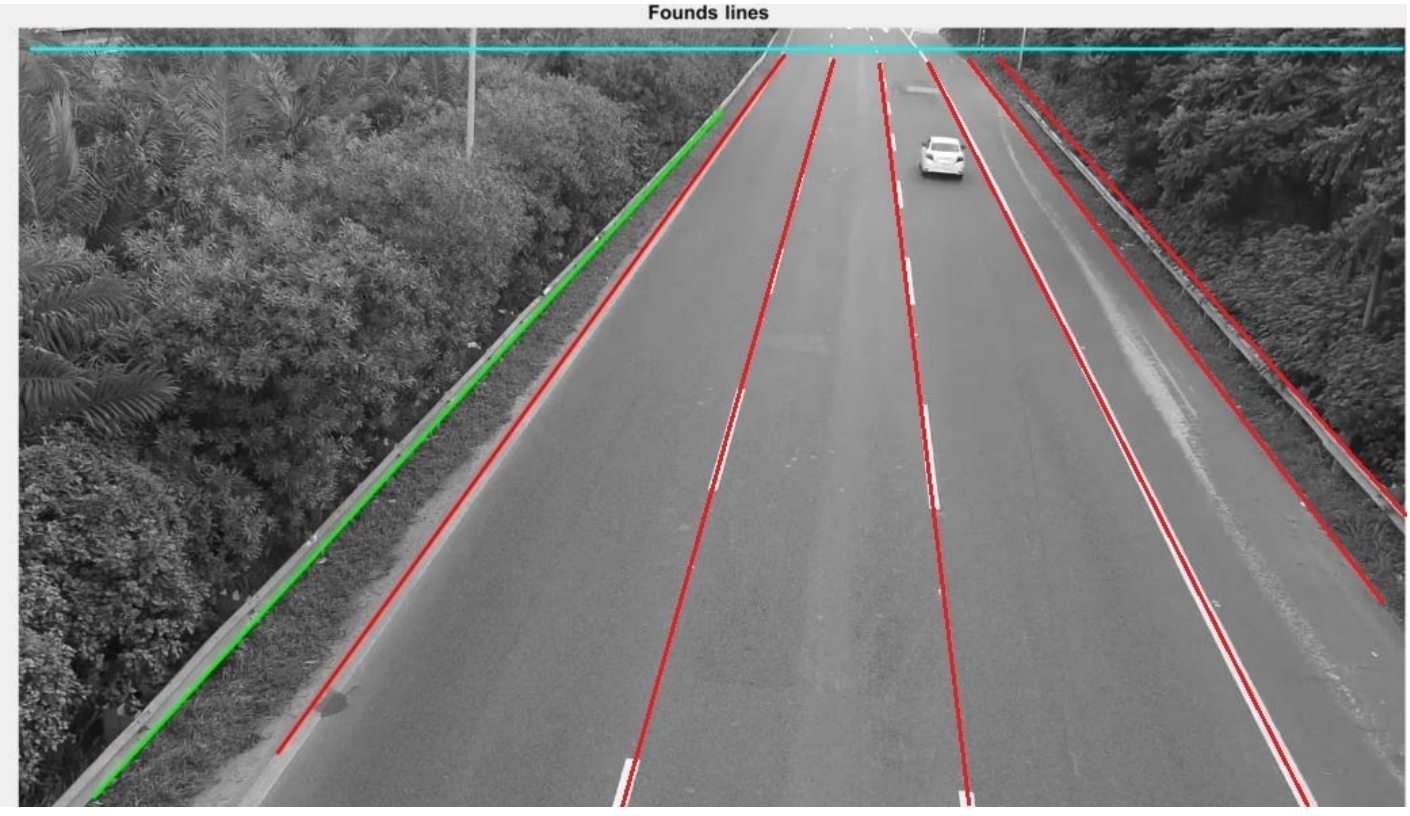
- HoughLines parameters are tuned to get better results on the detected lines.

- Related images with the line detection are given below:




Edges


Detected Hough Peaks
-1000
-500
0
500
1000
-80
-60
-40
-20
0
20
40
60
80
θ


Founds lines

- After the lines are found on the road image, boundaries (x and y coordinates from the image) of the lanes and road are stored in special variable as the calibration parameters.

- In addition to that special checkpoint from each line are marked and their locations are also stored as calibration parameters. These will be used for speed detection and movement direction detection.

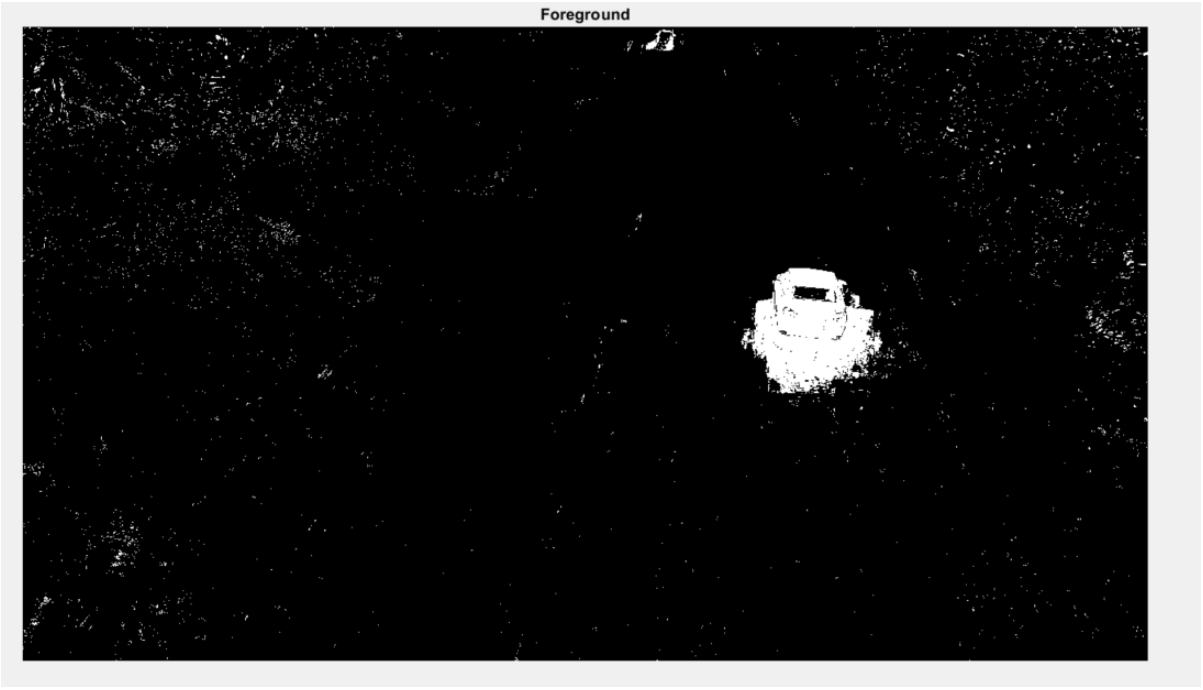
2. Foreground Detector

- For foreground detection a special tool in MATLAB called ForegroundDetector is used. This is a tool which can be trained by the different frames from the video and estimates the real foreground of the road.

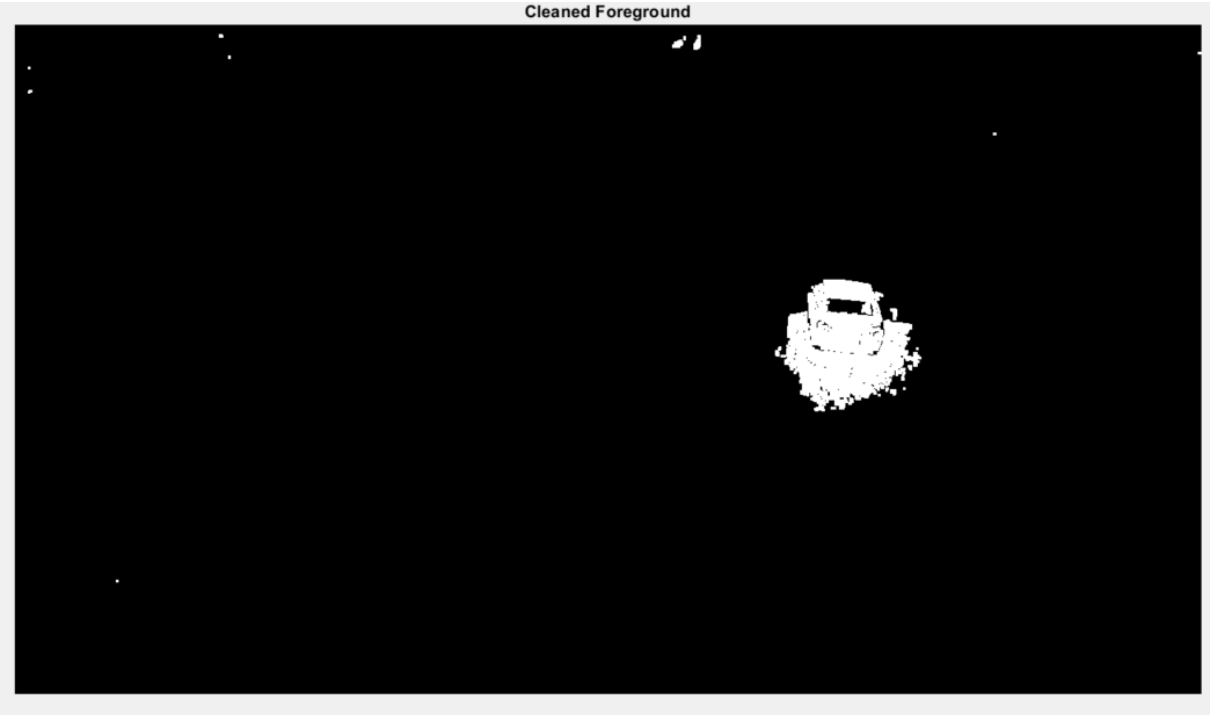
- This tool is trained with the videos and detect the foreground.

- In the training process, frames are given to the tool. However, these frames firstly processed with a Gaussian filter to be smoothed.

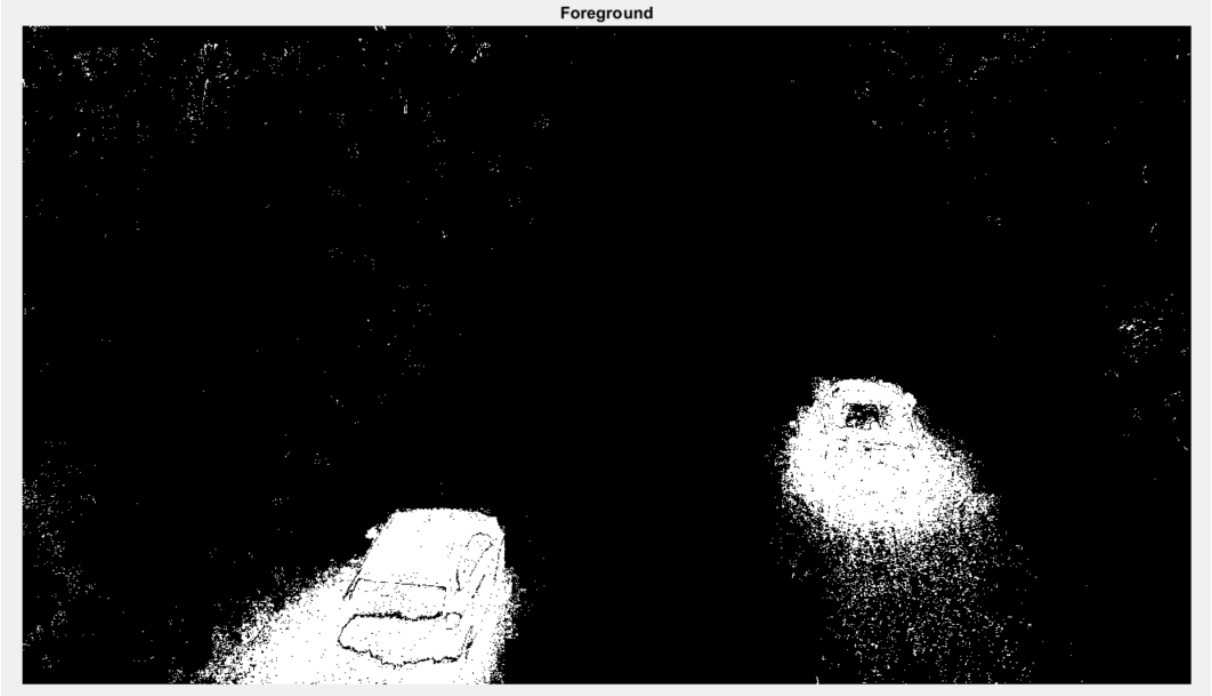
- Then it is tested with frames which includes moving objects on the road. In the images below, dark areas are estimated as the foreground and the white areas are estimates as the moving objects.

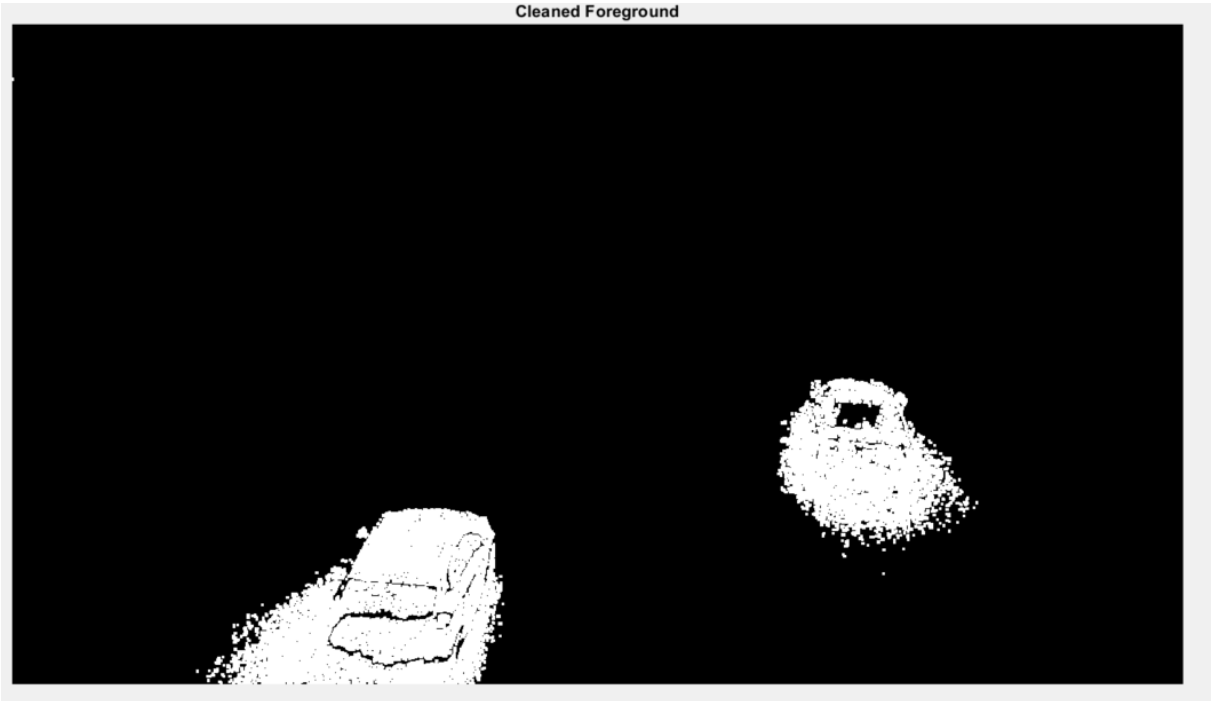


- Then this estimation image is processed to remove the noise. A special filter is used during this process. This filter is a morphological noise removal filter. A structuring element kernel is defined as square with size 3 for the filtering purposes. After this filter is applied to the image, noise is removed mostly but quality of the image is decreased because of the structuring element.



- These are other examples from the same process:





3. Moving object detection

- For this purpose, an analysis method called Blob analysis is used. This analysis technique can find an external object if the foreground is known. So, founded foreground in the previous step is given to the analysis tool. Then the tool gives the location and size of the external objects on the foreground.
- However, the output of the blob analysis was not efficient enough to use in the system. Therefore, we develop filter algorithms to increase the efficiency of the location of the moving objects.
- With this information, green boxes are drawn on the detected cars and their locations are stored for further analysis.

4. Determining the speed of objects

- Two different checkpoint areas in every lane are chosen in the previous steps.
- Also, the locations of the moving objects are also known with blob analysis.

- With using these special checkpoints and location of the moving objects, speed of the objects on each line is found.

- When a moving object passes the first checkpoint, a time counter is started. When the same object passes the second checkpoint, it starts and calculates the speed of the object.

- During this process, time calibration and road calibration is very important. Because the distance is calculated by the road calibration. Also, time is resized by the time coefficient to get better result. This time coefficient is found in the time calibration phase.

- In addition to that, direction of the movement is found with the same method. If a vehicle passes the second checkpoint before the first checkpoint, it is reported as the wrong way driving.

- Speed of every vehicle on every lane is calculated and printed on the top of the image for every lane.

5. Other violation detections

- Speed limit violation and occupancy of left lane in low speed is detected with the calculated speeds in the previous step.

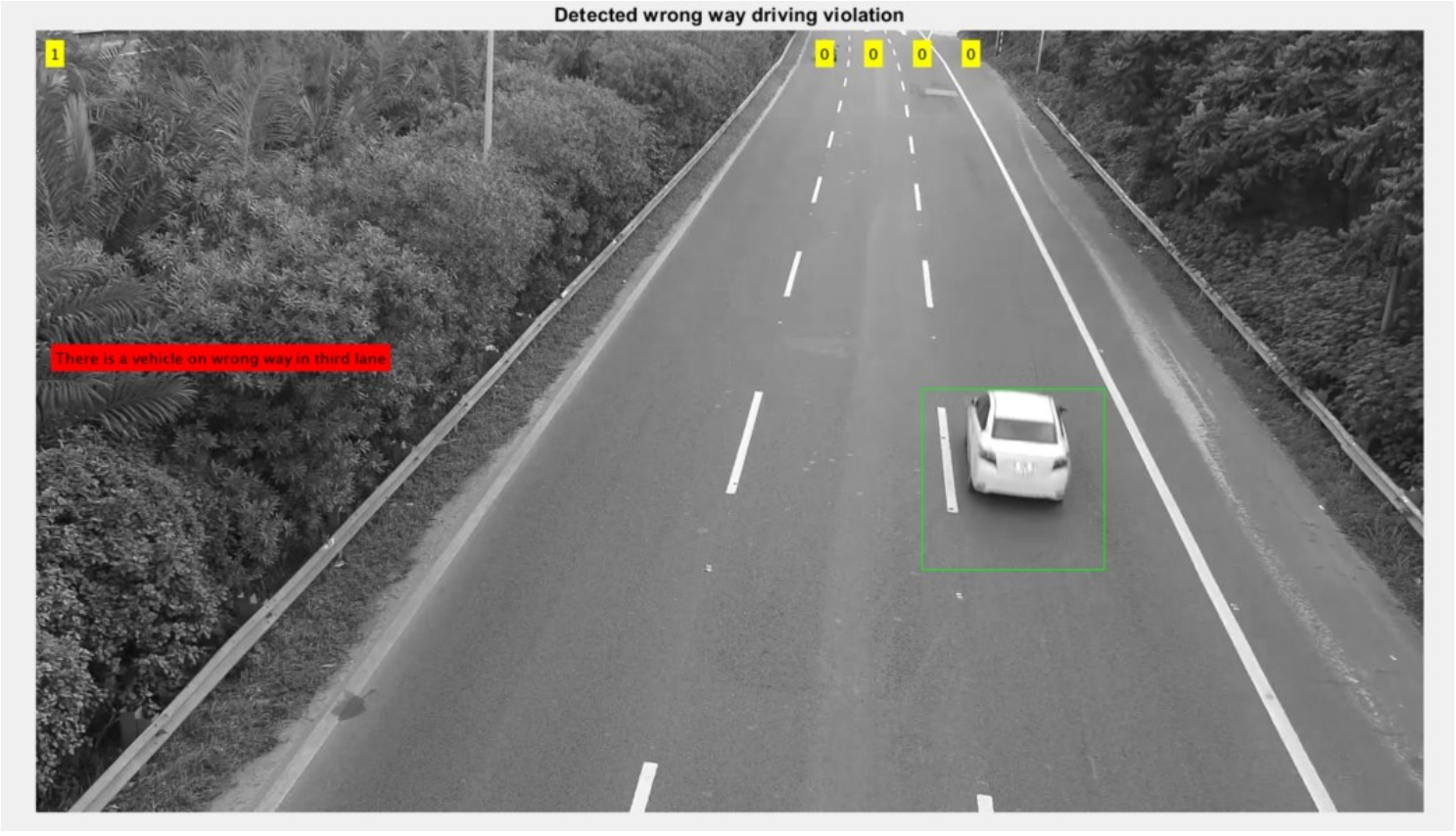
- In addition to that, because of the blob analysis and road calibration, location of every moving object is known. Also, location of the lanes is known. So, the lanes which vehicles drive is known. Therefore, if a vehicle moves on the breakdown lane, breakdown lane violation is detected.

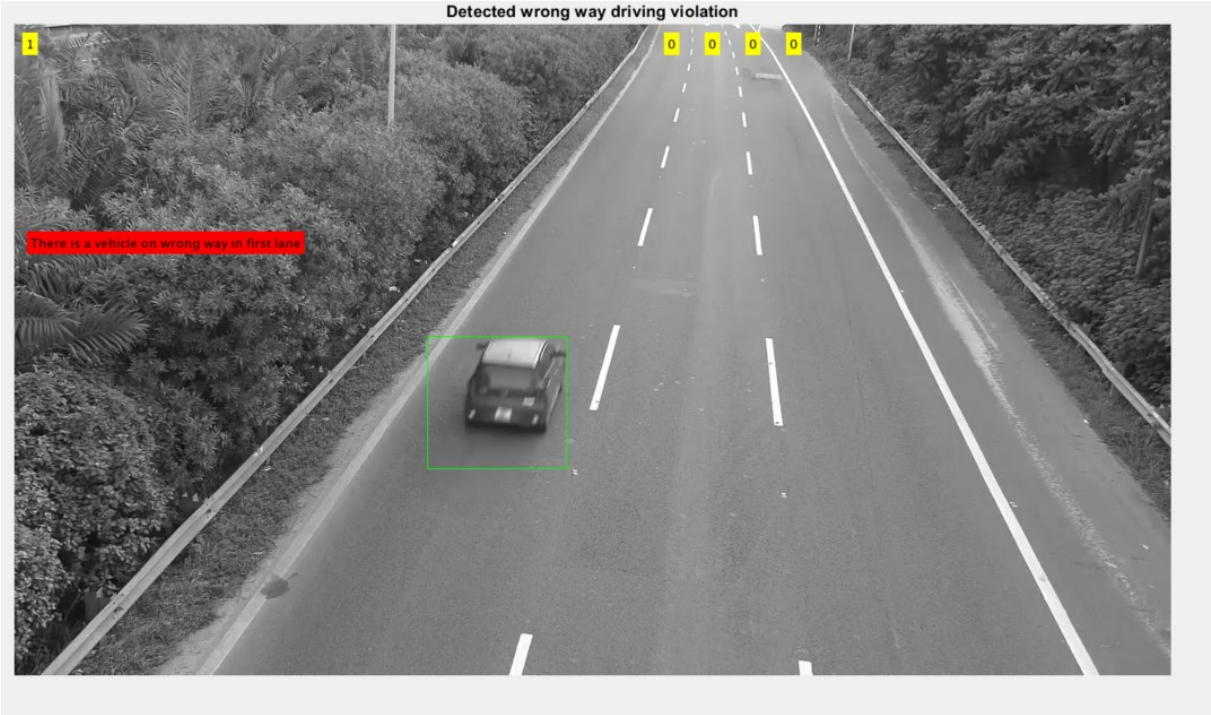
4. Results of the Project

Results of the implemented system is given below with the examples. In every example, a specific traffic violation is detected from the input video and reported by the system.

- **Wrong Way Driving Detection**

<u>Special Note:</u> For the wrong way driving detection, a suitable example video cannot be found. That's why a normal traffic video is reversed by the video tools and used for wrong way driving. (Video name = video5_grayscale)





- **Exceeded Speed Limit Detection**


Detected speed limit violation
3
113
104
54
54
There is a vehicle exceding speed limit in second lane


Detected speed limit violation
2
105
135
73
58
There is a vehicle exceding speed limit in first lane
There is a vehicle exceding speed limit in second lane

- **Breakdown Lane Violation**

- **High occupancy of left lane in low speed**

5. Discussion

To sum up, we did everything we planned. The results obtained are also quite sufficient. The lines were detected with houghlines and we separated the lanes with the obtained lines. Then we created the background of the frames with the ForeGround Detector we used. ForeGround Detector has been very useful for moving object detection. It doesn't work perfectly, but we didn't expect it to work perfectly either. It could detect unrelated objects as moving. We managed to solve this problem by constant training. The more the code runs, the better it understands the video and the better the results. Afterwards, we calculated the speed of moving objects according to their positions by keeping time. We also detected the movement in the opposite direction from the position difference in the frames. We printed the violations and warnings we identified on the screen. We also took pictures of these violations. As a result, we completed the project the way we wanted. For the later stages of the project, these filters can be developed or turned into a built-in function that gives speed and motion.

REFERENCE

[1] Buch, N., Velastin, S. A., & Orwell, J. (2011). A review of Computer Vision Techniques for the analysis of Urban Traffic. *IEEE Transactions on Intelligent Transportation Systems*, *12*(3), 920–939. https://doi.org/10.1109/tits.2011.2119372

[2] Pornpanomchai, C., & Kongkittisan, K. (2009, November). Vehicle speed detection system. In *2009 IEEE International Conference on Signal and Image Processing Applications* (pp. 135-139). IEEE.

[3] Sample video taken by: https://www.youtube.com/watch?v=QuUxHIVUoaY&t=1571s

Appendix

The main code for in-lab:

Editor - C:\Users\Kayra\Documents\MATLAB\project3\project_main.m

lab5houghlines.m | project_main.m | car_filter_center.m | car_filter.m

```
%%%%%%%%%%%%%%%    EE 417 COMPUTER VISION TERM PROJECT    %%%%%%%%%%%%%%%

%%%%%%%%%%%%%%    Detection of Basic Traffic Violations    %%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%    EFE ÖZTABAN 25202    %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%    KAYRA BİLGİN 25117    %%%%%%%%%%%%%%%%%%%%%%%%

%%
clear all; close all; clc;

%% Video Name

%video_name = "video3_grayscale.mov";

video_name = "video4_grayscale.mov";


%% Time Calibration

v = VideoReader(video_name);
numImgs = get(v, 'NumFrames');
duration = 12;

time_coefficient = duration/numImgs;
fprintf("Video has %5.1f number of frames. \n",numImgs);

fprintf("Time coefficient for the video is %5.5f. \n", time_coefficient);

%% Lane Calibration (Line Detection)

lab5houghlines(Img);

%% Lane Calibration (Variable Definitions) (for video4_grayscale)

line_lenght1 = 13;
line_lenght2=12;
line_lenght3=13;
line_lenght4= 14;

line1_y1 = 400;
line1_y2 = 430;

line1_lane1_x1 = 430;
line1_lane1_x2 = 640;

line1_lane2_x1 = 645;
line1_lane2_x2 = 840;
```

```
    line1_lane3_x1 = 845;
    line1_lane3_x2 = 1027;

    line1_lane4_x1 = 1038;
    line1_lane4_x2 = 1200;

    line2_y1 = 240;
    line2_y2 = 270;

    line2_lane1_x1 = 555;
    line2_lane1_x2 = 684;

    line2_lane2_x1 = 690;
    line2_lane2_x2 = 823;

    line2_lane3_x1 = 827;
    line2_lane3_x2 = 951;

    line2_lane4_x1 = 963;
    line2_lane4_x2 = 1070;

    %% ForeGround Detector Training

gaussian_num_for_training = 5;
num_of_repeated_traning = 5;

foregroundDetector = vision.ForegroundDetector('NumGaussians', gaussian_num_for_training, 'NumTrainingFrames', numImgs*num_of_repeated_traning );
videoReader = vision.VideoFileReader(video_name);


for j=1:num_of_repeated_traning
    for i = 1:numImgs

     frame = step(videoReader);                    % takes the next frame from the video
     frame = imgaussfilt(frame);                   % smooth the frame image with gaussian filter
     foreground = step(foregroundDetector, frame);  % trains the detector with the new frame

    end

    videoReader = vision.VideoFileReader(video_name);

end

%% Frame Processing Example Visulizations

expected_frame_num = 250;

```

```
for i = 1:expected_frame_num

    frame = step(videoReader);

end

figure
imshow(frame)            % frame image
title('Video Frame')


frame = imgaussfilt(frame);
foreground = step(foregroundDetector, frame);


figure
imshow(frame)            % smoothed frame image
title('Smoothed Video Frame')

figure
imshow(foreground)       % foreground image
title('Foreground')

structing_element = strel('square', 3);  % structing element for removing noise from the foreground
filteredForeground = imopen(foreground, structing_element);  % apply structring element of the foreground

figure;
imshow(filteredForeground);   % filtered foreground image
title('Cleaned Foreground');

%% Car Detection


carDetector_blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, 'AreaOutputPort', false, ...
'CentroidOutputPort', true, 'MinimumBlobArea', 150);

[centerPoints_of_cars,detected_car_boxes] = step(carDetector_blobAnalysis, filteredForeground);

result = insertShape(frame, 'Rectangle', detected_car_boxes, 'Color', 'green');
num_of_car_boxes = size(detected_car_boxes, 1);

result = insertText(result, [10 10], num_of_car_boxes, 'BoxOpacity', 1, 'FontSize', 14);

figure;
imshow(result);         % example detected cars image
title('Detected Cars');

%% Variable Definitions

speed_converter = 18/5;  % converts speed from m/s into km/h

speed_limit=100;

lower_speed_limit_left= 80;

error1 = 0;
error2 = 0;
error3 = 0;
error4 = 0;
error5 = 0;

error1_counter = 0;
error2_counter = 0;
error3_counter = 0;
error4_counter = 0;
error5_counter = 0;

error_message_timeout = 30;


text_message1= "There is a vehicle in the breakdown lane";
text_message2= "There is a vehicle exceding speed limit in first lane";
```

```
text_message3= "There is a vehicle exceding speed limit in second lane";
text_message4= "There is a vehicle exceding speed limit in second lane";
text_message5= "There is occupancy of the left lane by a vehicle in low speed";


%% Main Loop for Traffic Violation Detection

videoReader = vision.VideoFileReader(video_name);
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [950,400];

se = strel('square', 3);                          % morphological filter for noise removal

lane1_car_detector=1;
lane2_car_detector=1;
lane3_car_detector=1;
lane4_car_detector=1;

speed_of_lane1=0;
speed_of_lane2=0;
speed_of_lane3=0;
speed_of_lane4=0;

prev_speed_of_lane1 = 0;
prev_speed_of_lane2 = 0;
prev_speed_of_lane3 = 0;
prev_speed_of_lane4 = 0;
prev_speed_of_lane5 = 0;

frame_num=1;


while ~isDone(videoReader)

    tic

    frame = step(videoReader);                    % takes the next frame from the video
    foreground = step(foregroundDetector, frame); % smooth the frame image with gaussian filter
    filteredForeground = imopen(foreground, se);  % trains the detector with the new frame


    % Detect cars from the frame with filtered foreground
    [centerPoints_of_cars,detected_car_boxes] = step(carDetector_blobAnalysis, filteredForeground);
    center_of_detected_cars = step(carDetector_blobAnalysis, filteredForeground);


    % Filters the founded cars and boxes
     num = size(detected_car_boxes);

     if(num>1)
```

```
        detected_car_boxes = car_filter(detected_car_boxes,150);
        center_of_detected_cars = car_filter_center(center_of_detected_cars,150);
     end


    % Draw boxes around the detected cars
    result = insertShape(frame, 'Rectangle', detected_car_boxes, 'Color', 'green');
    [number_of_detected_cars,cc]=size(center_of_detected_cars);

    num_of_car_boxes = size(detected_car_boxes, 1);


    if(num_of_car_boxes>0)
        for i=1:number_of_detected_cars

            x_location = center_of_detected_cars(i,1);
            y_location = center_of_detected_cars(i,2);

            % first line detection for speed extraction

            if(y_location>line1_y1 && y_location<line1_y2)

                if x_location>line1_lane1_x1 && x_location<line1_lane1_x2
                    if(lane1 car detector==1)
                            a1=frame_num;
                            lane1_car_detector=0;
%                             disp("first check line 1")
                        end
                    end

                    if x_location>line1_lane2_x1 && x_location<line1_lane2_x2
                        if(lane2_car_detector==1)
                            a2=frame_num;
                            lane2_car_detector=0;
%                             disp("first check line 2")
                        end
                    end

                    if x_location>line1_lane3_x1 && x_location<line1_lane3_x2
                        if(lane3_car_detector==1)
                            a3=frame_num;
                            lane3_car_detector=0;
%                             disp("first check line 3")
                        end
                    end

                    if x_location>line1_lane4_x1 && x_location<line1_lane4_x2
                        if(lane4_car_detector==1)
                            a4=frame_num;
```

```
                        lane4_car_detector=0;

%                         disp("first check line 4")
                    end
                end

            end

            % second line detection for speed extraction

            if(y_location>line2_y1&&y_location<line2_y2)

                if x_location>line2_lane1_x1 && x_location<line2_lane1_x2
                    if(lane1_car_detector==0)

                        b1=frame_num;
                        t1=b1-a1;
                        t1 = t1*time_coefficient;

                        speed_of_lane1=line_lenght1/t1;
                        speed_of_lane1=speed_of_lane1*speed_converter;

                        lane1_car_detector=1;

%                         disp("second check line 1")
```

```
                        end
                    end

                    if x_location >line2_lane2_x1 && x_location<line2_lane2_x2
                        if(lane2_car_detector==0)

                            b2=frame_num;
                            t2=b2-a2;
                            t2 = t2*time_coefficient;

                            speed_of_lane2=line_lenght2/t2;
                            speed_of_lane2=speed_of_lane2*speed_converter;

                            lane2_car_detector=1;
%                             disp("second check line 2")

                        end
                    end

                    if x_location>line2_lane3_x1 &&x_location<line2_lane3_x2
                        if(lane3_car_detector==0)

                            b3=frame_num;
                            t3=b3-a3;
                            t3 = t3*time_coefficient;
```

```
322
323 -                           speed_of_lane3=line_lenght3/t3;
324 -                           speed_of_lane3=speed_of_lane3*speed_converter;
325
326 -                           lane3_car_detector=1;
327     %                         disp("second check line 3")
328
329 -                       end
330 -                   end
331
332
333 -                   if x_location>line2_lane4_x1 &&x_location<line2_lane4_x2
334 -                       if(lane4_car_detector==0)
335
336 -                           b4=frame_num;
337 -                           t4=b4-a4;
338 -                           t4 = t4*time_coefficient;
339
340 -                           speed_of_lane4=line_lenght4/t4;
341 -                           speed_of_lane4=speed_of_lane4*speed_converter;
342
343 -                           lane4_car_detector=1;
344
345     %                         disp("second check line 4")
346
```

```
347 -                       end
348 -                   end
349
350 -               end
351
352 -           end
353 -       end
354
355
356 -       num_of_car_boxes = size(detected_car_boxes, 1);
357
358
359       % display the results on the frame image
360
361 -     result = insertText(result, [10 10], num_of_car_boxes, 'BoxOpacity', 1, 'FontSize', 14);
362 -     result = insertText(result, [720 10], round(speed_of_lane1), 'BoxOpacity', 1, 'FontSize', 14);
363 -     result = insertText(result, [765 10], round(speed_of_lane2), 'BoxOpacity', 1, 'FontSize', 14);
364 -     result = insertText(result, [810 10], round(speed_of_lane3), 'BoxOpacity', 1, 'FontSize', 14);
365 -     result = insertText(result, [855 10], round(speed_of_lane4), 'BoxOpacity', 1, 'FontSize', 14);
366
367
368       % error message displays
369 -     if(error1 == 1)
370
```

```
%         disp(text_message1)
        result = insertText(result, [15 80], text_message1, 'BoxOpacity', 1, 'FontSize', 14,'BoxColor','red');
        error1_counter = error1_counter+1;

        if (error1_counter == error_message_timeout)
            error1 = 0;
        end

    end

    if(error2 == 1)

%         disp(text_message2)
        result = insertText(result, [15 110], text_message2, 'BoxOpacity', 1, 'FontSize', 14,'BoxColor','red');
        error2_counte = error2_counte+1;

        if (error2_counte == error_message_timeout)
            error2 = 0;
        end

    end

    if(error3 == 1)

```

```
%         disp(text_message3)
        result = insertText(result, [15 140], text_message3, 'BoxOpacity', 1, 'FontSize', 14,'BoxColor','red');
        error3_counte = error3_counte+1;

        if (error3_counte == error_message_timeout)
            error3 = 0;
        end

    end

    if(error4 == 1)

%         disp(text_message4)
        result = insertText(result, [15 170], text_message4, 'BoxOpacity', 1, 'FontSize', 14,'BoxColor','red');
        error4_counter = error4_counter+1;

        if (error4_counter == error_message_timeout)
            error4 = 0;
        end

    end

    if(error5 == 1)

```

```
%         disp(text_message5)
        result = insertText(result, [15 200], text_message5, 'BoxOpacity', 1, 'FontSize', 14,'BoxColor','yellow');
        error5_counte = error5_counte+1;

        if (error5_counte == error_message_timeout)
            error5 = 0;
        end

    end

    % error detectors

    if(lane4_car_detector == 0)
        if(error1 == 0)

            error1 = 1;
            error1_counter = 0;

        end
    end

    if(speed_of_lane1>speed_limit)

        if(error2 == 0 && speed_of_lane1 ~= prev_speed_of_lane1)
```

```
443
444 -                error2 = 1;
445 -                error2_counte = 0;
446 -                prev_speed_of_lane1 = speed_of_lane1;
447
448 -            end
449 -        end
450
451 -        if(speed_of_lane2>speed_limit)
452
453 -            if(error3 == 0 && speed_of_lane2 ~= prev_speed_of_lane2)
454
455 -                error3 = 1;
456 -                error3_counte = 0;
457 -                prev_speed_of_lane2 = speed_of_lane2;
458
459 -            end
460 -         end
461
462 -        if(speed_of_lane3>speed_limit)
463
464 -            if(error4 == 0 && speed_of_lane3 ~= prev_speed_of_lane3)
465
466 -                error4 = 1;
467 -                error4_counter = 0;
```

```
468 -                prev_speed_of_lane3 = speed_of_lane3;
469
470 -            end
471 -        end
472
473 -        if(speed_of_lane1<lower_speed_limit_left)
474
475 -            if(error5 == 0 && speed_of_lane1 ~= prev_speed_of_lane5)
476
477 -                error5 = 1;
478 -                error5_counte = 0;
479 -                prev_speed_of_lane5 = speed_of_lane1;
480
481 -            end
482 -        end
483
484
485 -    step(videoPlayer, result); % display the result on the result video
486
487 -    frame_num = frame_num+1;
488
489
490 -    end
491
492
493 -    release(videoReader); % close the video file
```

lab5houghlines.m | project_main.m | car_filter_center.m | car_filter.m

```
function filtered_data = car_filter_center(data,threshold)

filtered_data = data(1,:);
[row,col] = size(data);

for i=2:row

    check = true;
    [num1,num2] = size(filtered_data);

    for j=1:num1

        if(abs(data(i,1)-filtered_data(j,1)) < threshold)
            if(abs(data(i,2)-filtered_data(j,2)) < threshold)
                check = false;
            end
        end


    end

    if(check == true)
        filtered_data = [ filtered_data; data(i,:) ];
    end
end
end
```

lab5houghlines.m | project_main.m | car_filter_center.m | car_filter.m

```
function filtered_data = car_filter(data,threshold)

filtered_data = data(1,:);
[row,col] = size(data);
data = sortrows(data,3,'descend');

for i=2:row

    check = true;
    [num1,num2] = size(filtered_data);

    for j=1:num1

        if(abs(data(i,1)-filtered_data(j,1)) < threshold)
            if(abs(data(i,2)-filtered_data(j,2)) < threshold)
                check = false;
            end
        end
    end

    if(check == true)
        filtered_data = [ filtered_data; data(i,:) ];
    end
end
end
```