PROJECT REPORT

CAROLO CUP 2015

TEST AND INTEGRATION TEAM

SAULIUS EIDUKAS

MALLIGARAJ MALLESWARAN

STEFANIA FERNANDEZ

YOHANNES G. TESFAGIORGIS

June 15, 2015

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden

Abstract

Carolo Cup is a student competition for designing and building an autonomous miniature vehicle. The competition contains different disciplines namely lane following, parking and obstacle avoidance to test various components of the vehicle. This report summarizes the work of Test and Integration group of team MOOSE that participated in Carolo Cup 2015 from Chalmers University of Technology. The report explains the testing mechanism used on the recordings of the track to find the performance of algorithms used in different disciplines of the competition. Further a test catalogue containing all the test scenarios is given. An evaluation of the existing algorithm is performed to find its strengths and weaknesses.

Contents

1 Introduction

Carolo Cup is an international student competition organized by Braunschweig University of Technology in Braunschweig, Germany. The aim of the competition is to design and develop an autonomous miniature vehicle. The model vehicle is a 1:10 scale car that has to drive around a scaled down track with the help of camera and other sensors. The challenge is to realize the best possible vehicle control in different scenarios inspired by realistic environment. The teams compete in various disciplines for a total of 1000 points. The winner is decided by a jury of experts and academia based on the total number of points that each team obtains in different disciplines. The competition does not impose any restrictions on the hardware or software environments that should be used [1].

2 Competition

The objective of the competition is that the student team is hired by a fictious car manufacturer to develop, produce and demonstrate a possible cost-effective and energy-efficient concept of autonomous vehicle. The competion consists of the following two disciplines.

2.1 Static Discipline

The static discipline consists of the presentation of manufacturing, energy balance and description of the various design decisions made by the team. This static discipline comprises of 35% of the overall points. It is a twenty minutes presentation of various features of the car and ways in which the team handled different components of the competition.

2.2 Dynamic Discipline

The dynamic discipline evaluates the capabilities of the autonomous model vehicle. It consists of three parts namely,

2.2.1 Parallel Parking

While driving on the right lane, the vehicle has to find a parking spot on its right hand side and park in as fast as possible without colliding with other vehicles (mimicked by obstacles). The challenge is that at least three wheels of the car should be inside the parking strip. The team loses points when the car touches any of the obstacles or touches the driving lane. The car uses the Infrared LEDs to measure the distance between two obstacles and decide if the gap is long enough for the car to park and also used the Ultrasonic sensors to avoid touching the obstacles.

2.2.2 Course without Obstacles

The car has to follow lanes of the track using the camera to detect the lanes of the road. The track is made up of a non-reflecting black surface over which the lane markings are done in white. The left and right lanes have a fixed width of 20 mm (usually 19 mm which is the width of a regular electrical tape). The middle lane markings have a width of 20 mm, length of 20 cm and also have an interval of 20 cm between two consecutive lane markings. The width of the intersection marking is 40 mm. The radius of the curves should not be less than 1m avoiding very sharp curves. The complexity of this part of the competition is increased in the competition by the removal of lane markings at different parts of the track. The lane markings can be missing for a maximum of 1m at any part of the road except intersection. The car has to follow the lane without going out of the lane or crossing the middle lanes for two minutes. Points are given based on the distance travelled by the car in two minutes and the team loses points every time the car goes out of the lane or when a person takes control of the car on occurrence of misbehaviour.

2.2.3 Course with Obstacles

This part is to test the car's ability to overtake obstacles. The car has to follow the lanes normally until it encounters an obstacle. The car has to move to the other lane, overtake the obstacle and move back to its original lane. It should also be able to do the same with the moving obstacle. Then the car has to wait and allow a moving obstacle on the other side of the intersection.

3 Testing

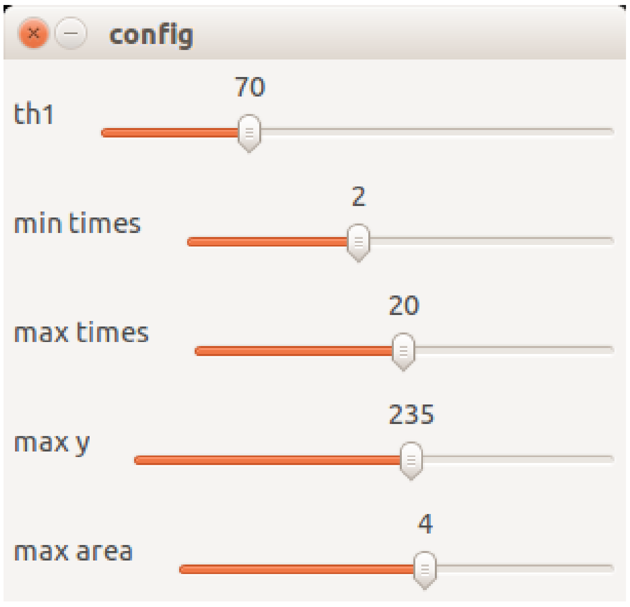
This section describes the procedure and the factors involved in testing the different algorithms used in the project. The task of the testing team is to make the car go through different testing scenarios and provide feedback to development teams. One way of doing this is to make the car drive around and note down the places on the track where the it fails. One obvious drawback of this approach is that we will only know where the car fails and it is quite difficult to find out why the car fails. It is also time consuming, as it should be done for every change the development team makes. In order to overcome that a recording of the full track is done and the algorithm is run on the recording.

3.1 Recording Videos

The team used the OpenDaVINCI framework for the development of algorithms. This framework has a very useful feature of video recording. So the camera on the car can be used to record the track and then the algorithm can be run on the recording to see the track from the car's perspective. In this way it is easy to find why the car failed, as we will see the exact frame that led to the failure of the car. So changes can be made in the algorithm to improve the case in which the car failed.

3.1.1 Calibration

It is important to calibrate the camera to its optimal level to obtain a clear video. It also helps in meeting the quality of image expected by the algorithm. The window used for calibrating the camera is shown in Figure 1.


config

th1 70

min times 2

max times 20

max y 235

max area 4

Figure 1: Camera Calibration Window

The first step is to calibrate the camera to obtain a sharp image. It can be done by changing the focus of the camera until a clear image is obtained and also adjusting the polarizing filter to remove maximum reflection. Then the *threshold* parameter has to be adjusted until all the lines are detected in the output window while running *lanedetector* component. The *max times* parameter has to be adjusted until the dashed lines on the road are detected properly. A low *max y* value makes the dashed lines to be detected as solid lines, so an optimal value has to be set to that so that the dashed lines are detected correctly and the *max area* parameter helps in detecting the intersection correctly. A detailed explanation of tweaking each parameter is given in [2].

3.2 Manual Inspection

It is a procedure in which the algorithm is applied to each frame of the recording and checked for errors in the frame. The main aim of the algorithm is to analyze each image and detect lines which in our case correspond to the right lane, left lane and the middle lane markings. So depending on the position and angle in which these lines are found the car maneuvers around the track. So the role of testing is to check how these lines are detected by the algorithm. The decision made by the car for maneuvering around the track based on these images can be tested only by dríving the car around the track. An example of the manual inspection window is shown in Figure 2.

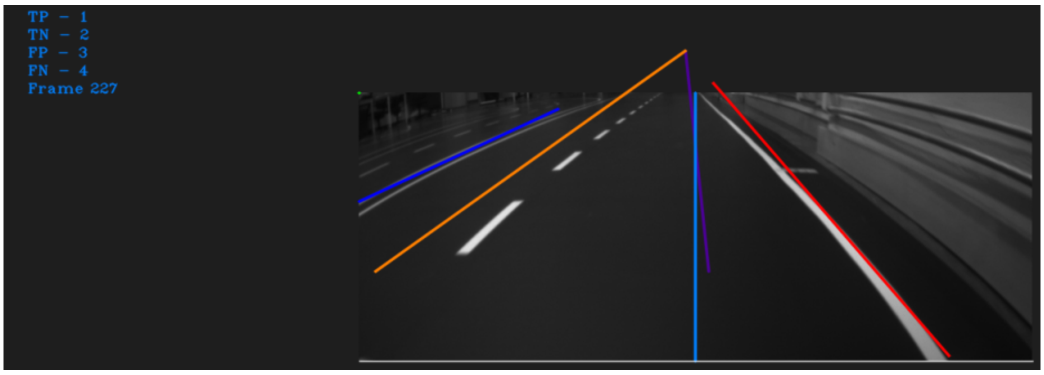


Figure 2: Manual Inspection View

The procedure for carrying out manual inspection is to categorize each frame into one of the four cases depending on the output of the algorithm on each frame. As seen on the top left corner of Figure 2, a frame is categorized into any of the four categories by pressing the corresponding numbers. The key presses for each frame is stored in a *.csv* file in the same folder that contains the recordings.

The categorization of the frames are listed below,

3.2.1 True Positive

A frame is categorized as true positive if,

- At least one of the dashed middle lines that are present in the frame is detected and highlighted.
- Right lane is present and is highlighted.
- Left lane is present and is highlighted.

An example of a True Positive frame is shown in Figure 3.

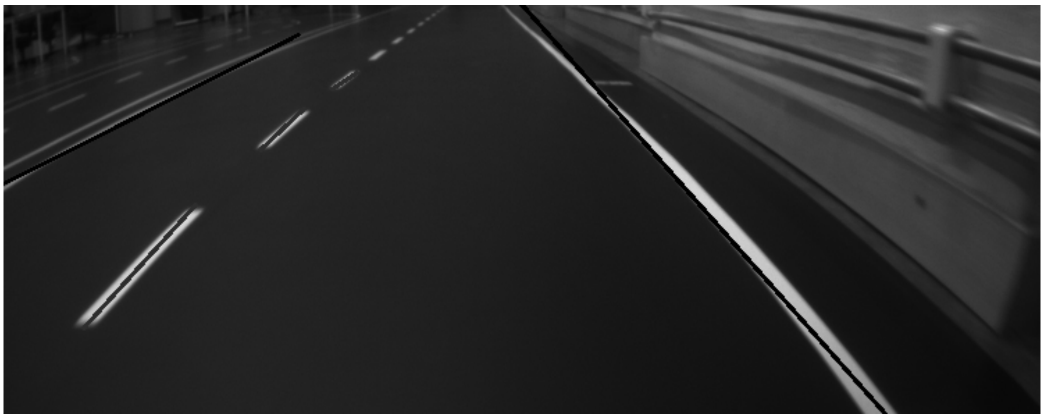


Figure 3: Example of a True Positive Frame

In 3 the left lane, right lane and three dashed lines are detected and highlighted by the algorithm. Since it satisfies the condition it is categorized as True Positive frame. Other examples of True Positive frames are shown in Figures 4 & 5.

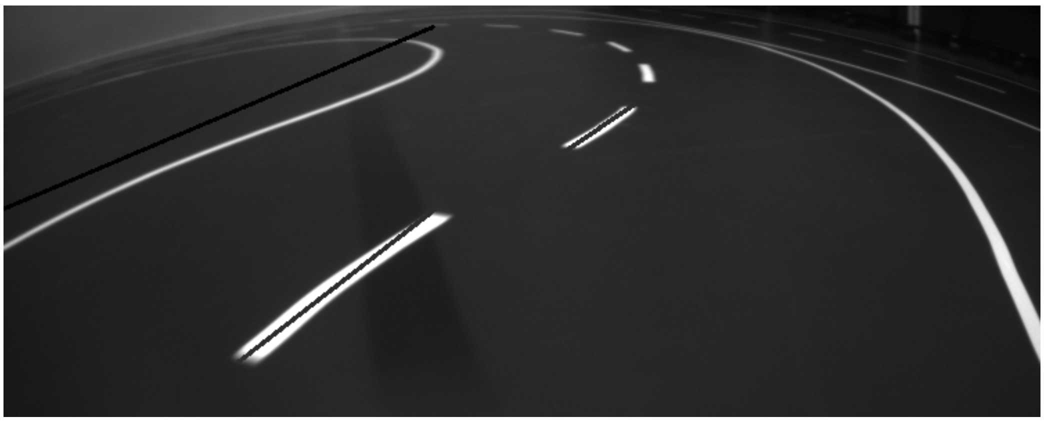


Figure 4: Example of a True Positive Frame

The dashed lines and left lane is detected and still satisfies the condition for True Positive.

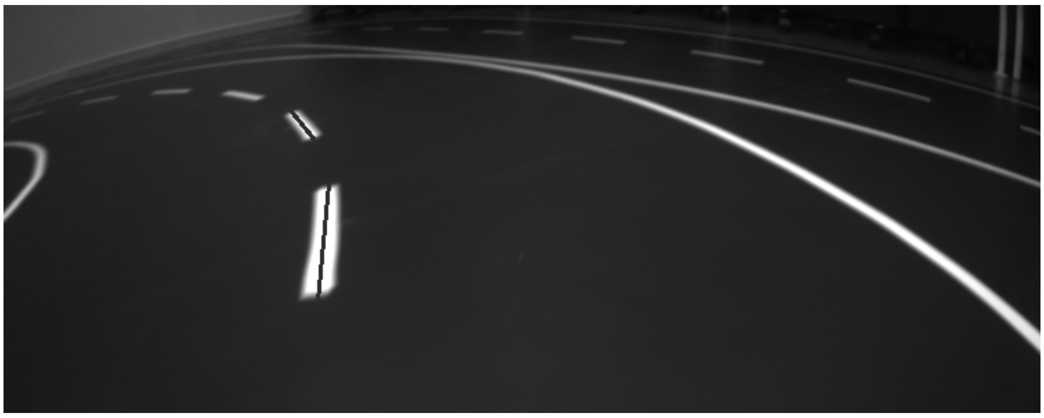


Figure 5: Example of a True Positive Frame

Only dashed lines are detected and qualify for a true positive frame.

3.2.2 False Negative

A frame is categorized as false negative if at least one lane marking is present but is not highlighted by the algorithm. An example of False Negative is shown in Figure 6.

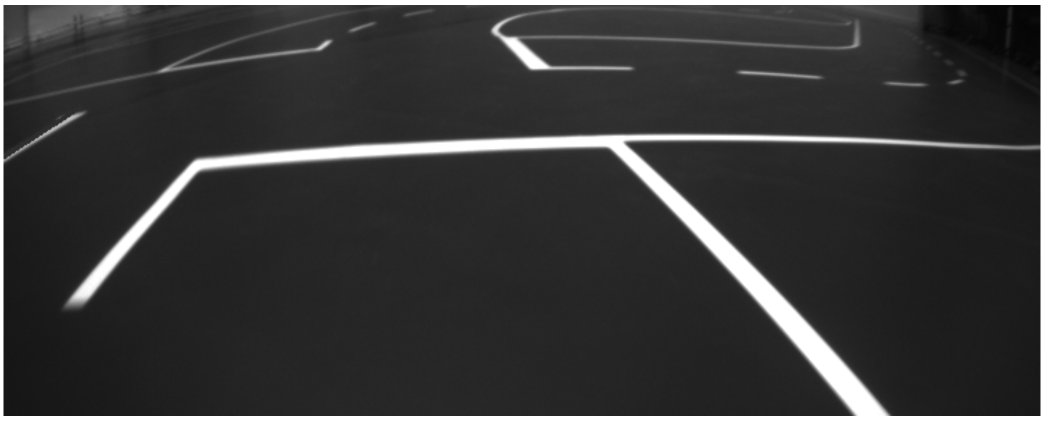


Figure 6: Example of a False Negative Frame

Although there are lanes around the car, the algorithm does not detect any lines making it a False Negative Frame.

3.2.3 False Positive

A frame is categorized as False Positive if there are no lane markings in the frame but something is highlighted by the algorithm. An example of False positive frame is shown in Figure 7.

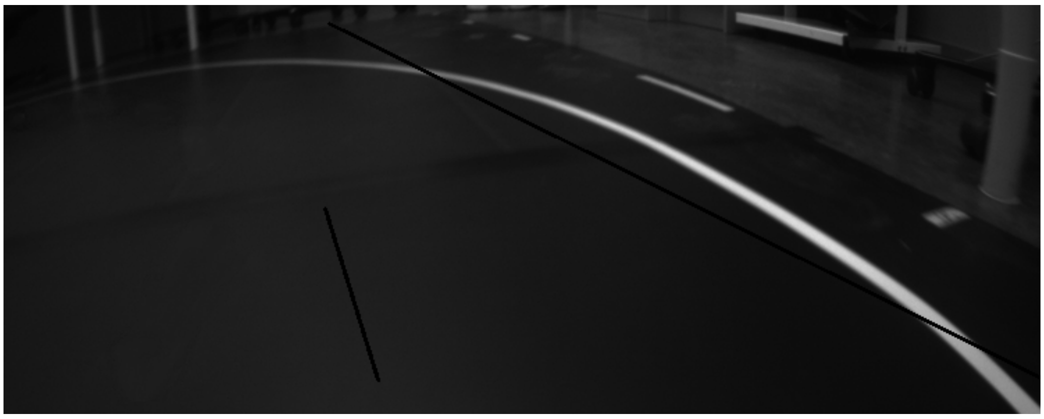


Figure 7: Example of a False Positive Frame

In Figure 7, there is no dashed line in the frame, but the algorithm highlights something making it a False positive frame.

3.2.4 True Negative

A frame is categorized as True Negative if there are no lane markings in the frame and the algorithm does not highlight anything. An example of True negative is shown in Figure 8.

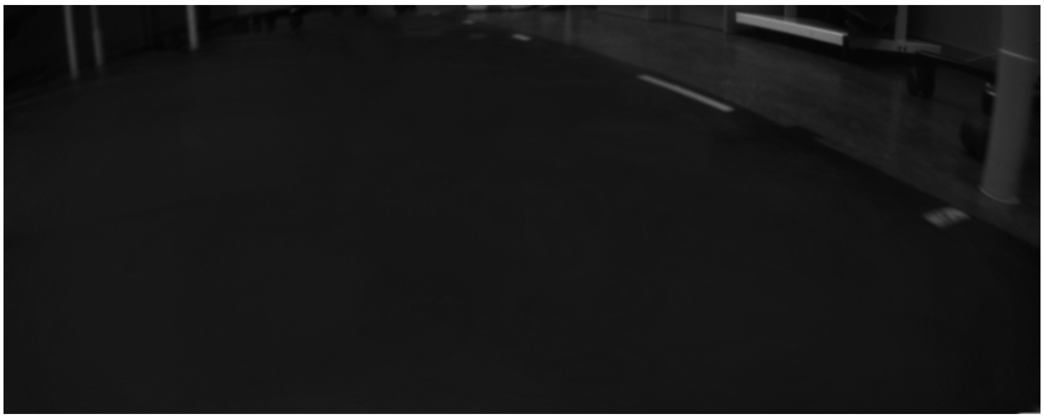


Figure 8: Example of a True Negative Frame

In Figure 8, there are no lane markings and the algorithm does not highlight anything making it a True Negative frame.

3.3 Automated Inspection

Manual inspection is laborious and time consuming, as the performance of every change in the algorithm has to be checked manually for all the frames and is also inclined to human errors. A solution to that would be to automate the inspection process. This is implemented using a Python script that performs the necessary steps to inspect all the recordings present in the target directory. Before the start of the automated inspection every new recording has to be manually inspected to obtain the ground truth (*.csv*) file.

The local copy of the files are updated after a change in the algorithm code. Then the script can be used to start the automated inspection. It first compiles the local copy of the project and exits on occurrence of any errors during compilation. The *csvextractor* component is run on successful completion of compilation, that generates a *.csv* file corresponding to every recording present in the target folder. The generated file contains information like frame number, co-ordinates of left lane, co-ordinates of right lane, co-ordinates of dashed line, co-ordinates of goal line etc. for each frame. Now it finds the differences that the changes in the algorithm code created. It is done by finding the distance between a left lane in a frame to the left lane of the corresponding lane in the ground truth file using the co-ordinates present. In the same way it finds the distances for right lane, dashed line and goal line. These distances are then compared to a threshold that was framed by visually inspecting the distances to see if the lines detected are still acceptable. The calculation of threshold has to be done manually once for every new recording. Any frame that violates this threshold is saved in a separate file.

4 Test Catalogue

This section explains the concept of Test catalogue and explains the use of it in this project. The test catalogue is a collection or an index of all the different test scenarios that can be performed to check the different aspects of the algorithm. These test scenarios are performed on the recordings of the track. The recordings of the current track are available in the network drive located at `https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/`. A general structure to keep track of different tests is shown in Table 1.

Test Number	
Attempt Number	
Description of the Test	
Result of the Test	
Passed/Failed	
Date/Person in-charge	

Table 1: Structure to organize different tests

A list of tests for Lane following, Lane detection and Parking are given below,

4.1 Lane Following

The flow, expectation, failure of scenario, location of recording and the corresponding ground truth files for different test scenarios are given below.

4.1.1 Drive on a Straight Road

The test to check the performance of the algorithm on a straight road is given in Table 2.

Basic Flow	The car is put on a straight road on the right lane and *lanedetector* is initiated. Car starts driving forward on the straight road up to 6m.
Expectation	The car drives straight without changing wheel angles.
Scenario Failure	Car goes outside its lane. Car keeps changing wheel angle.
Location	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/StraightRoad`

Table 2: Test to check the performance of algorithm in Straight Road

4.1.2 Drive in a Right Curve

The test to check the performance of the algorithm on a Right curve is given in Table 3.

Basic Flow	The car is placed in the entrance of a right curve. *lanedetector* is True Positive (Both lanes are detected). Car's wheels turn right. Car goes through the right curve.
Expectation	The car drives through the curve with high stability.
Alternative Flow	The car is placed in the entrance of a right curve. Only one of the lanes is detected by *lanedetector*. Car's wheels turn right. Car goes through the right curve.
Scenario Failure	*lanedetector* does not detect lanes (True Negative). Car does not turn right and keeps driving straight. Car under-steers to the right. Car over-steers to the right.
Location	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/RightCurve`

Table 3: Test to check the performance of algorithm in a Right Curve

4.1.3 Drive in a Left Curve

The test to check the performance of the algorithm on a Left curve is given in Table 4.

Basic Flow	The car is placed in the entrance of a left curve. *lanedetector* is True Positive (both lanes are detected). Car's wheels turn left. Car goes through the curve.
Expectation	The car drives through the left curve with high stability.
Alternative Flow	The car is placed in the entrance of a left curve. Only one of the lanes is detected by *lanedetector*. Car's wheels turn left. Car goes through the curve.
Scenario Failure	*lanedetector* does not detect lanes (True Negative). Car does not turn left and keeps driving straight. Car under-steers to the left. Car over-steers to the left.
Location	`https://se-div-c3s-1.ce.chalmers.se:5006/` `carolocup/TestCatalogue/LaneDetector/LeftCurve`

Table 4: Test to check the performance of algorithm in a Left Curve

4.1.4 Drive in a S-Curve

The test to check the performance of the algorithm on a S-curve is given in Table 5.

Basic Flow	Car is placed in the entrance of a S-Curve. *lanedetector* is True Positive (both lanes are detected). Car's wheel turn left. Car goes through left curve. Car enters right curve. Car's wheels turn right. Car goes through right curve.
Expectation	The car drives through the S-Curve with high stability.
Alternative Flow	Car is placed in the entrance of a S-Curve. Only one of the lanes is detected by *lanedetector*. Car's wheel turn left. Car goes through left curve. Car enters right curve. Car's wheels turn right. Car goes through right curve.
Scenario Failure	*lanedetector* does not detect any lanes (True Negative). Car does not turn left and keeps driving straight. Car under-steers to the left and goes out of the lane. Car over-steers to the left and goes out of the lane. Car under-steers to the right and goes out of the lane. Car over-steers to the right and goes out of the lane.
Location	`https://se-div-c3s-1.ce.chalmers.se:5006/` `carolocup/TestCatalogue/LaneDetector/SCurve`

Table 5: Test to check the performance of algorithm in a S-Curve

4.1.5 Drive in a Straight Road Close to Another Road

The test to check the performance of the algorithm on a Straight road close to another road is given in Table 6.

Basic Flow	Car is placed in a straight road next to another road section on right lane. *lanedetector* is True Positive (both lanes are detected) Car drives forward on the straight road up to 6m.
Expectation	The car drives straight without changing wheel angles.
Scenario Failure	*lanedetector* does not detect any lanes (True Negative). Car picks up the other road and follows it. Car goes out of the lane. Car keeps changing wheel angle.
Location	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/RoadNextRoad`

Table 6: Test to check the performance of algorithm in a Straight Road next to Another Road

In addition to above test scenarios few additional recordings with its ground truth can be found in locations shown in Table 7.

Full Track	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/FullTrack`
Intersection	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/Intersection`
Missing Lanes	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/MissingLanes`
Obstacles	`https://se-div-c3s-1.ce.chalmers.se:5006/carolocup/TestCatalogue/LaneDetector/Obstacles`

Table 7: Location of Additional Recordings

4.2 Partial Tests for Lane Detection Algorithm

A set of tests that can be performed for checking the performance of the lane detection algorithm is given below.

4.2.1 Contour Detection

The test to check the quality of Contour detection is given in Table 8.

Description	Test if all the wanted lines have at least one corresponding rectangle.
Expectation	All wanted lines have at least one corresponding rectangle.
Procedure	Compare the raw frame with a frame with all the rectangles added.

Table 8: Test to Check Contour Detection

4.2.2 Lines Classification

The test that can be performed to check the classification of lines by the algorithm is given in Table 9.

Description	Test if the rectangles from corresponding lines are correctly interpreted as solid line, dashed line, stop line or an intersection.
Expectation	All wanted lines are correctly interpreted and no false interpretation is made.
Procedure	Compare a frame with all the rectangles added and a frame with all the classified lines.

Table 9: Test to check Line Classification

4.2.3 Lines Filtering

The test that can be performed to check the filtering of unwanted lines by the algorithm is given in Table 10.

Description	Test if all the lines except the wanted lines are filtered away.
Expectation	Only the wanted lines are remaining after the filtering.
Procedure	Compare a frame with all the classified lines and a frame with the lines remaining after the filtering.

Table 10: Test to check Filtering of Lines

4.2.4 Lines Usage

The test that can be performed to check if the right lines are used for lane following by the algorithm is given in Table 11.

Description	Test if the correct/best found lines are used when calculating the goal and vanishing point.
Expectation	Only the correct/best found lines are used when calculating the goal point and vanishing point.
Procedure	Compare a frame with the lines remaining after the filtering and a frame displaying the used lines.

Table 11: Test to check Usage of Lines

4.2.5 Estimation of Undetected Lines

The test that can be performed to check the estimation of undetected lines is given in Table 12.

Description	Test the estimation of undetected wanted lines that are needed for calculating goal and vanishing point.
Expectation	If a detected wanted line is removed, the estimate should be reasonably close to the present line.
Procedure	Start with a line remaining after filtering, which serves as the goal. Remove the detected line and run it through the estimation function (that has to be created) and compare the result with the goal.

Table 12: Test to check Estimation of Undetected Lines

4.3 Hardware Test Scenarios

In addition to the above mentioned tests for the software, few of the tests that can be used to check the hardware is given below.

4.3.1 Sensor Connectivity

The test that can be performed to check the connectivity of sensor is given in Table 13.

Description	Test if all sensors are connected, powered and running.
Expectation	Each sensor should be powered and provide data
Procedure	An Arduino code that reads data from each sensor can be used. The data might be displayed on the LCD screen present on the car or serial monitor present in the Arduino IDE.

Table 13: Test to check Sensor Connectivity

4.3.2 Sensor Data

The test that can be performed to check the correctness of the sensor data is given in Table 14.

Description	Test if the sensors are providing accurate information.
Expectation	The sensors should provide correct data with difference in accuracy that does not affect the behavior of the car.
Procedure	Manual monitoring of individual sensor data can be done.

Table 14: Test to Check Sensor Data

4.4 Parking

The test that can be used to check the parking scenario is given in Table 15.

Basic Flow	Place the car parallel to the parking strip and initiate parking sequence. Car drives straight measuring the parking spot lengths. Car finds a parking spot big enough and parks the car.
Expectation	The car should park correctly without touching any obstacles or crossing the parking strip.
Scenario Failure	Car fails to initiate the parking sequence and drives straight ahead. Car hits an obstacle. Car crosses the parking strip and any of the wheels touch the driving lane.

Table 15: Test to check Parking Scenario

5 Evaluation

This section shows the performance of the current algorithm. This evaluation considers only the detection of lanes from the frame and the decisions that are made by car to drive based on these frames are not evaluated. The performance of the algorithm is evaluated from the ground truth files obtained, as mentioned in Section 3. The performance of the algorithm in different scenarios such as Full track, Straight road, Curves, Missing Lanes are measured. It is measured by the number frames that are classified as True Positive, False Negative, False Positive and True Negative.

5.1 Full Track

A manual inspection of the full track provided the performance values as shown in Table 16.

Total Frames	1653
True Positive	1584
False Negative	69
False Positive	0
True Negative	0

Table 16: Frame Categorization of Full Track

From Table 16, it can be seen that 95.82% of the frames are classified as True Positive. This number is directly proportional to success rate of the algorithm. The recording is started only after the car is placed on the track, which means there are lane markings surrounding the car at all times during the recording. So the conditions False Positive and True Negative do not occur. Further it can be seen that 4.18% of the frames that are categorized as False Negative are mainly due to the intersections and start box (which can be observed from the ground truth file).

5.2 Straight Road

The categorization of frames on a straight road is given in Table 17. These values include the tests with Straight road and Straight road close to another road.

Total Frames	1615
True Positive	1614
False Negative	0
False Positive	0
True Negative	1

Table 17: Frame Categorization of Straight Road

From the values in Table 17, it can be seen that the algorithm performs perfectly in straight roads.

5.3 Curves

The categorization of frames on curves is given in Table 18. These values include the test cases with Right curve, Left curve and S-curve.

Total Frames	2064
True Positive	2063
False Negative	0
False Positive	0
True Negative	1

Table 18: Frame Categorization of Straight Road

From Table 18, it can be seen that the algorithm works really good for curves as well.

5.4 Missing Lanes

The categorization of the frames with missing lane marking of 50 cm long on Right side or Dashed lines is given in Table 19.

Total Frames	899
True Positive	897
False Negative	2
False Positive	0
True Negative	0

Table 19: Frame Categorization of Full Track

From Table 19, it can be seen that the algorithm performs extremely well as almost all the frames are classified as True Positive. This is mainly due to the fact that the car can drive correctly on the lanes with the detection of just a single dashed line. However this high performance can be achieved only with proper calibration as explained in Section 3.1.1. Again since the recording is started only after the car is placed on the track, there are lanes surrounding the car at all times during the recording.

6 Discussion

This section provides a discussion about strengths and weaknesses of the algorithm from evaluation results. The performance of different aspects of the project and few suggestions for improving the current setup are also given.

6.1 Lane Following

The evaluation presented above shows the performance of the algorithm on the recordings of the track. Careful calibration of the camera and good lighting conditions were maintained to achieve the best possible recording with clear images. From the evaluation results it can be seen that the algorithm performs extremely well in straight roads, curves and even with missing lanes. Even though the evaluation results show very good numbers for the algorithm, there are other factors that affect the performance of the car.

6.1.1 Hardware Constraints

The reaction time or responsiveness of the car on the track depends on the number of frames successfully processed in unit time. In addition to that, there is a real-time deadline associated with every frame. A decision on trajectory that the car should follow has to be made for the frame before the deadline. For the car to follow lanes successfully at low speed (for example 0.5 m/s) the hardware processor has to process around 30 frames per second (at 33 ms per frame). The frames per second to be processed increases with the speed of the car. So when the car is run at slightly higher speeds, the deadlines for making the decisions are reduced and are sometimes missed. Due to this the car momentarily loses its trajectory and moves away from the ideal path. The situation becomes worse when several deadlines are missed and the car goes out of its trajectory making it impossible to retrieve control. This leads to the car going out of the lane or degraded stability.

6.1.2 Lighting and Reflection

The environment around the car should be optimum to attain the maximum performance. The first step in the algorithm after receiving an image is to apply binary thresholding to remove noise. The value of threshold to be used for this is set based on the lighting using the calibration procedure explained in Section 3.1.1. Selection of optimal threshold value is important for proper highlighting of the lanes. So any change in lighting conditions or improper selection of threshold value leads to improper highlighting of the lanes, which in turn impacts the performance and stability of the car. An example output image after thresholding with optimal threshold value is shown in Figure 9. This issue was improved by using a light sensor, that allows dynamic setting for threshold value according to the lighting. There was a significant improvement in the performance of the car after the inclusion of light sensor.

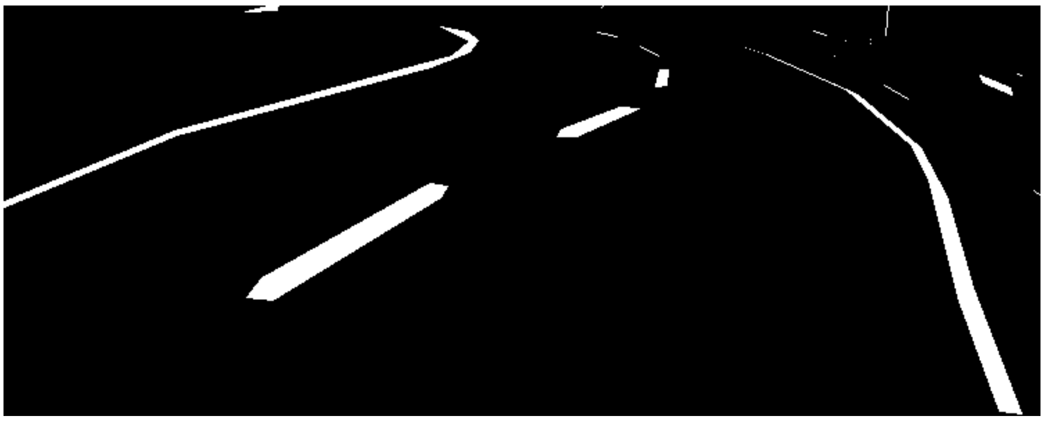


Figure 9: Example Image after Thresholding

The quality of images are affected by reflections from the track. Polarizing filter is placed in front of camera lens to remove reflections or suppress glare from the surface of the track. Incorrect tuning of this filter allows light patches on the image due to reflection leading to improper output from binary thresholding stage. Wrong decisions on trajectory of the car is made due to improper detection of lanes.

6.1.3 Camera Calibration

The algorithm detects the lanes by finding contours of different objects present in frames. The calibration of the camera has to be done periodically after a few test runs. Improper tuning of camera parameters lead to the generation of blurry images making the cotouring process inaccurate. This leads to reduction in performance of the car.

6.1.4 Intersection

The evaluation results show that the algorithm does not handle intersection properly. An example of such a case in which the algorithm has failed at intersection is shown in Figure 10.

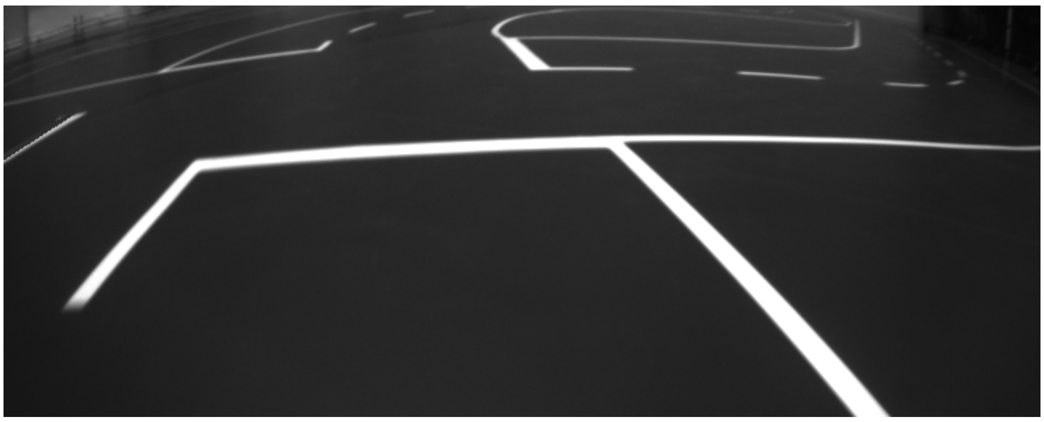


Figure 10: Output from Final Filter

From Figure 10, it can be seen that the algorithm has resulted in a False Negative case leading to failure of the system. As mentioned above the algorithm detects lanes from the frame by finding contours of different objects. The output of such contours fitted in a rectangle is shown in Figure 11.

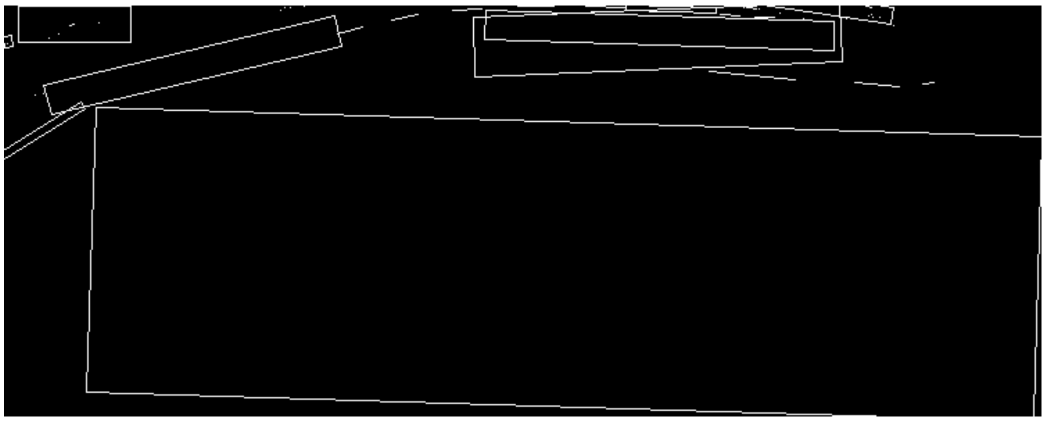


Figure 11: Output from Contours fitted in Rectangles

In Figure 11, the large rectangle in the bottom of the frame represents the stop line of intersection. The shape and angle of this rectangle is dependent on the angle at which the car approaches the stop line. If it does not approach the stop line with the ideal right angle approach (where the stop line and car are perpendicular to each other), it is difficult for the car to recognize it as an intersection. As seen in Figure 11, since the car did not approach in right angle with the stop line, the lanes that are present around stop line are detected as a right curve. Due to this the car considers itself to be present in a right curve and makes it. This problem should be solved by either improving the existing algorithm or making a new algorithm to handle the intersections.

6.1.5 Speed of Car

In addition to above improvements the speed of the car could be improved as well. In competition and practice runs the car could successfully find lanes and drive around at a speed of 0.8 m/s to 1.2 m/s and this is definitely not enough for winning the competition. So the speed of the car could be increased by finding an alternative hardware or optimizing the algorithm to make it work faster.

6.2 Automated Inspection

The existing version of the automated inspection is an initial version that can be further improved. To start with it can check for any changes in the github repository and pull the changes automatically. Further it can be made to automatically send mails to the authors of the code, regarding any error that might occur during compilation and the results of inspection.

6.3 Test Catalogue

The current test catalogue covers all the tests that would check the basic functionalities of the lane detection algorithm. The test catalogue can be improved by recording more test scenarios to increase the quality and robustness of the lane following and parking algorithms. But since the current algorithm fails the most in Intersections, it is important to create test scenarios focusing just the intersections. So a few recordings of the car driving in and out of intersections from all four sides can be added.

The current recordings with missing lanes have lanes missing up to 50 cm, bus according to competition rules the lanes can be missing up to 1 m. Further these missing lanes can be in Straight roads or curves. A few suggestions are given in Table 20. Each case on the right column can be recorded for every possible scenario in the left column.

Straight Road Right Curve Left Curve S-Curve	Missing Right lane for 1 m
	Missing Left lane for 1 m
	Missing Middle lane for 1m
	Missing multiple Right lanes of 50 cm each
	Missing multiple Left lanes of 50 cm each
	Missing multiple Middle lanes of 50 cm each
	Missing Right and Left lanes of 1 m each
	Missing Right and Middle lanes of 1 m each
	Missing Middle and Left lanes of 1 m each
	Missing multiple Right and Left lanes of 50 cm each
	Missing multiple Right and Middle lanes of 50 cm each
	Missing Middle and Left lanes of 50 cm each

Table 20: Suggestions for Recordings with Missing lanes

In addition to above recordings to test the lane detection, a few recordings to test the obstacle avoidance can also be added. According to competition rules, stationary or mobile obstacles can be present anywhere on the track. A few suggestions for recording with obstacles is given in Table 21.

Straight Road Right Curve Left Curve S-Curve	Obstacle on Right lane
	Obstacle on Left lane
	Multiple obstacles on Right lane
	Multiple obstacles on Left lane
	Obstacle on Right and Left lane 1 m apart
	Multiple obstacles on Right and Left lanes 1 m apart from each other
Intersection	Obstacle on the adjacent intersection in the right side

Table 21: Suggestions for Recordings with Obstacles

6.4 Parking

The car was able to qualify for parking during the competition by doing a very good parallel parking in between two obstacles. But it was not able to reproduce that during the competition costing 100 points for the team. So a reliable algorithm should be developed for it. The parking algorithm does not use the lane following for initiating the parking, instead it just drives forward until it a spot and then initiates the reverse S sequence. This driving forward mechanism is so unreliable due to bad alignment of wheels and wrong placement of the car on track. So it is better to integrate the lane following, so that the car follows the lane until it finds a suitable parking spot.

6.5 Obstacle Avoidance

In the current version of the software, only a basic version of obstacle avoidance is available and this was done as a last resort solution to just qualify for that discipline. The current version uses the Ultrasonic sensors in the front to detect an obstacle and the distance from it. So either the same hardware configuration can be used with a better algorithm to make it work or an entirely new sensor arrangement can be used. One suggestion for a new sensor would be to use Light detection and Ranging (LIDAR) or Camera to detect the object in front of the car and make decisions based on it.

Just to share a few final notes on this project, it is important to have a working version of the car after every change in the software. Good team organization and communication is the key in such projects. Some kind of social network in which every one in the team can be contacted immediately is necessary.

7 Future Work

So to summarize the future work that could be done to improve the car are,

- Improve algorithm for handling Intersection.
- Improve algorithm for handling Start box.
- Increase the speed of the car (supported by both hardware and software).
- Improve the automated inspection procedure.
- Integrate lane following into parking.
- The reliability of parking algorithm should be improved (may be by extensive testing).
- Improve the algorithm for obstacle avoidance.

8 Conclusion

This report summarizes the work of the Testing and Integration team for the Carolo Cup 2015. The ability to record the track and test the algorithm on the recording is a very useful feature of OpenDaVINCI platform. It was very helpful to make the testing faster, easier and efficient. It played an important role in the competition, as the access to the track in Germany was time limited. As mentioned before the Test Catalogue can also be improved by adding more test scenarios.

References

[1] "Carolo Cup - Rules and Regulations 2015," Carolo Cup 2015.

[2] Calibration Procedure. github. [Accessed: 22 January 2015]. [Online]. Available: https://github.com/se-research/CaroloCup/wiki/Calibration-procedure