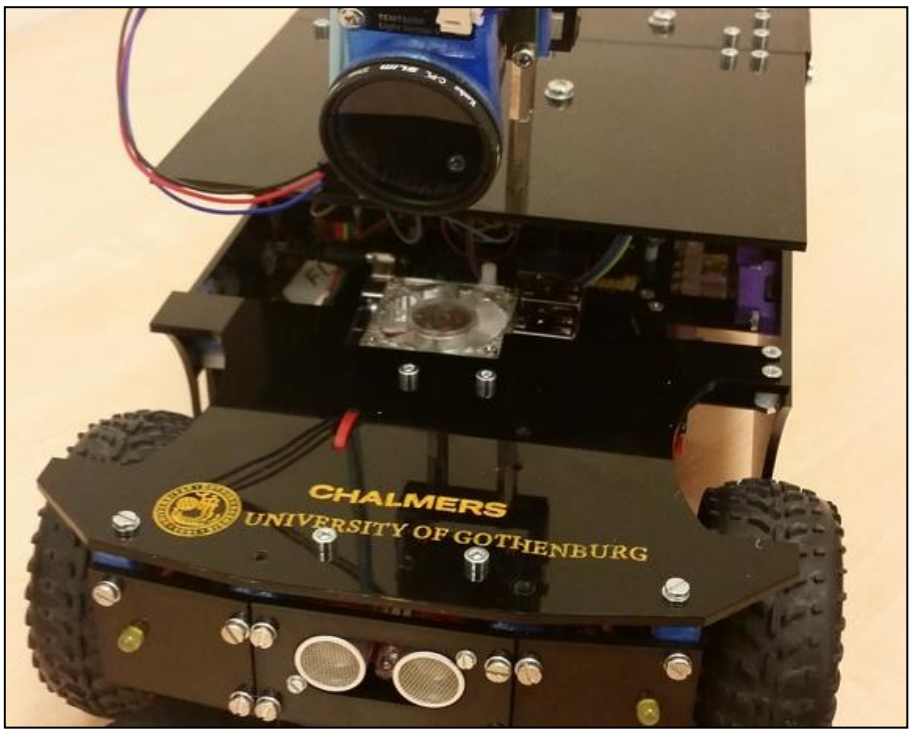
UNIVERSITY OF GOTHENBURG



Lane Detection and Following Approach in Self-Driving Miniature Vehicles

Bachelor of Science Thesis in Software Engineering and Management

IBTISSAM KAROUACH
SIMEON IVANOV

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

Lane Detection and Following Approach in Self-Driving Miniature Vehicles

IBTISSAM KAROUACH
SIMEON IVANOV

Academic Supervisor: CHRISTIAN BERGER
Examiner: JAN-PHILIPP STEGHÖFER

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2016

Lane Detection and Following Approach in Self-Driving Miniature Vehicles

Ibtissam Karouach
University of Gothenburg
Software Engineering & Management BSc
karouach.ibtissam@gmail.com

Simeon Ivanov
University of Gothenburg
Software Engineering & Management BSc
simeon@ivanovsimeon.com

***Abstract*—The current work, discusses the various experiences, lessons learned and developed novelties, regarding the lane detection and lane following features of miniature autonomous vehicles. Particularly, we outline the work that was conducted, during our team's participation in Carolo Cup 2016, which is an international competition on autonomous vehicle, held in Braunschweig, Germany. An overview of selected works from the relevant literature is presented, in order to illustrate the various techniques that are commonly adopted in order to implement the lane detection and following functions. The findings, in this regard, constitute of a set of the mostly referenced methods. Additionally, our adopted approach on the matter is extensively elucidated. The different improvements that were introduced to the existing algorithm, are quantified and visualized through an evaluation tool that was developed for this purpose. We conclude that the use of the said tool for analysis, increases the quality of the final product, which is of significant importance in embedded system, due to the fusion of software and hardware elements.**

I. Introduction

Self-driving vehicles are expected to outnumber conventional vehicles by 2050, most of them capable of autonomous driving at all times. Fully self-driving vehicles projected to hit widespread adoption by 2035 [1][2]. Society of Automotive Engineers (SAE) classifies full automation as vehicles able to (a) complete a journey from point A to point B without any input from the driver beyond setting the destination, (b) drive at least as well as an average person, on any road, adhering to all the traffic laws of its time and (c) handle any extreme situation without the driver taking over, thus foregoing the need to include any manual controls, i.e. a wheel and pedals. Complete independent driving is designated to Level 5 on the self-driving scale, proposed by SAE, as opposed to Level 0 where the driver needs be in constant control, but can still have computer aid in form of warnings, e.g. collision warning system, lane departure warning system [3].

One of the earliest and most widespread adaptation of lane detection is in lane departure warning systems. The component alerts the driver when the car starts steering outside the lane, be it for the reason of drivers getting distracted or sleep deprivation. It is especially useful on highways where the monotony of driving may exacerbate inattention. According to a study conducted for the International Journal of Injury Control and Safety Promotion, trucks are responsible for 11% of fatal road accidents, in USA alone, a major cause being sleep deprivation [4].

In the future, the widespread adoption of autonomous vehicles is expected to reduce traffic accidents by a factor of 10, as well as greatly reducing the numbers of vehicles on the streets [5]. Other potential benefits include: increased leisure time and productivity during commute [6], higher speed limit [7] and removing restrictions for vulnerable groups [8].

Research on development of automated vehicles, as we envision them today, has been conducted as early as 1980s, Eureka PROMETHEUS Project [9] and DARPA's ALV being the pioneers of the decade [10]. DARPA's ALV has successfully demonstrated the first road following using lidar and computer vision [10]. By 1989, the ALV has also introduced machine learning to their repertoire [11]. In 2004, 2005 and 2007, DARPA has organised Grand Challenge competitions in desert and urban environments, giving students and academia researches the opportunity to develop solutions for traversing aforementioned scenarios and win prizes [12]. Close collaboration between academia, the public sector and the private sector remains to this day.

DARPA's Grand Challenge is not the only competition of its kind, Carolo Cup offers similar challenges, but at a smaller scale. The goal of the competition is to build a 1/10 scale model self-driving car that is able to follow lane-markings, overtake obstacles, pass intersections safely and park inside a sideways parking strip autonomously [13]. The benefits of operating miniature vehicles is they are substantially cheaper and faster to build, lowering the entry level for students participating in similar competitions and eases prototyping in R&D. Student competitions are a good venue to gather talent for the ever-growing autonomous vehicles industry. It introduces the students to the field and instills confidence in their ability to contribute, this may lead to them being interested in considering the venture as a career path as well. It also helps attracting the attention of the public which may in turn inspire the new generations to pick studies related to the field, entice new private investments as well as allow for more government funding.

In 2016, we have participated in the Carolo Cup competition held in Braunschweig, Germany, representing Sweden. Every year the competition is gaining a wider spectrum, this year, almost 600 people assisted to watch the finals and, for the first time, has also been televised. Our team, Team Pegasus, has taken the 10th place, receiving the least penalty points in the lane following discipline. Our experience and the data

gathered during development serves as the basis of this study.

A. Problem Domain and Motivation

As mentioned previously, the most appealing feature of self-driving cars is the ability to transport passengers and/or cargo, from point A to point B, autonomously in a safe manner. In order to achieve this goal, the vehicle should have some form of road following system, traversing through both rural and busy urban streets while abiding all the existing traffic laws. Two common ways for a car to follow a road, without modifying existing roads or attaching various sensors on other cars, is to either use GPS tracking or machine vision. According to the U.S. government, the accuracy of a GPS tracking unit is approximately 7.8m at 95% confidence level [14]. Lanes on average are between 2-4m width, so the accuracy of a GPS is not good enough to keep a car within its lane reliably. The other approach, machine vision, would rely on lane markings and other road features extracted from the footage of an on-board camera to enable lane detection and subsequent following. However, lane markings may not be always clearly defined, the camera view may get obstructed by on-road traffic or other obstacles and various weather and light conditions could affect the visibility of the camera [15]. Nevertheless, this approach provides the most accurate horizontal position of a car within a road and with well-thought algorithms and hardware addons, the aforementioned problems can be alleviated or downright eliminated. Ideally, you would use both methods to get to set point B, machine vision and other sensors to keep the car on its lane and avoid obstacles, and GPS tracking to identify the vehicle's position in relation to the set final destination and derive a path to it.

In Carolo Cup, one of the main requirements is to be able to follow the lanes of a previously unmet and unmapped road, the participating teams are not informed of the layout of the road beforehand. This outright eliminates GPS tracking. As the participating vehicles are 1/10 size of real cars, the course track is miniaturised accordingly so the inaccuracy of GPS is magnified by factor of 10. Therefore, machine vision remains the sole solution. The competition track is arranged as a circuit and the aim of the lane following discipline is to complete as many laps as possible in the allotted time, there is no need for a car to go to a specific location in the track. The lane following discipline also includes intersection scenarios, missing lane markings and obstacles on the road. The parking discipline may also require lane following for the purpose of keeping the car in the middle of its lane.

Lane Detection	Extract and identify observed lane markings correctly.
Lane Following	Derive trajectory based on the extracted lane markings or other approximations. The trajectory should lead to the middle of the lane the car is located in.

TABLE I
TAXONOMY

We define lane detection as the ability to extract and identify observed lane markings correctly, i.e. right line, dashed line, left line. Lane following is deriving the trajectory for the car to aim for, e.g. pointing towards the middle of its lane, based on the lane markings detected previously [Table I]. Normally, the trajectory is derived on a 2D plane, i.e. in the camera footage, cars operate in a 3D environment so the trajectory has to be translated accordingly. The translation depends upon the camera used and its position in relation to the road as each pixel in the image has to be mapped to each point in the real world to find out what distance each pixel represents, e.g. 1 pixel = 10cm. To ensure a smooth transition of the car to desired trajectory, i.e. the car does not overshoot its desired destination, a control loop of some kind has to be implemented, e.g. a PID control. Tuning the translation calculation is out of scope of this study as it is dependent upon each equipments'

B. Research Goal & Research Questions

The goal of this study is to research and analyze various implementations of lane detection and following in miniature vehicles. Its initial purpose is to introduce the reader to the subject matter. Later, the research outcomes are used to supplement our own findings in implementing lane detection and following in a miniature self-driving vehicle for the Carolo Cup competition of 2016.

Considering the aforementioned, the following research questions are proposed:

RQ-1: What strategies are commonly adopted for successful lane detection?

RQ-2: How does adding an extract road phase affect the precision, accuracy, and processing time of a lane detection and following algorithm?

C. Contribution

The contributions of this study are (a) the outcome of a literature review listing common techniques in achieving lane detection and following in vehicles, and (b) an analysis and showcase of a lane detection and following implementation in a miniature vehicle with a demonstration of our proposed improvements.

D. Scope

The main purpose of this study is to demonstrate a lane detection and following algorithm developed to overcome the challenges presented in the Carolo Cup competition of 2016. Carolo Cup regulations describe the minimum requirements for vehicles to qualify, road size and layouts and planned scenarios. Generally, these scenarios are less complex than what full-sized self-driving vehicles are expected to encounter in day to day life, e.g. constant light conditions, no weather hazards, less stationary and moving objects, single type of intersection etc. The competition is restricted to students only, the vehicles are 1/10 scale of a regular car and are generally made of low-cost components. Despite addressing common techniques of lane detection and following in all vehicles, the literature review is to be tailored towards selecting methods that are applicable in low-cost miniature vehicles.

E. Structure of The Article

The next section describes the methodologies we have adopted in order to answer the established research questions. The background section introduces the reader to machine vision concepts, the controlled environment of the Carolo Cup competition and the tools available in the OpenDaVINCI framework that can help in the development of a lane detection and following module. In the related work section, we have outlined the common lane detection techniques we have gathered from literature review and the presentation slides of the teams who participated in the Carolo Cup competition of 2016. Next, we describe in detail a lane detection and following algorithm developed by team MOOSE who participated in the 2015th edition of Carolo Cup. Then, we showcase the tools we have developed in order to benchmark the performance of the algorithm and present the results in form of graphs. In the next section, we present our own improvements of the lane detection algorithm introduced previously. Finally, we discuss and analyze our benchmarking results, our contribution to the improvement of the lane detection algorithm, and possible faults of this study.

II. Methodology

A. Research Question I

To address RQ1, we have conducted a literature review of related work to identify common techniques used in achieving lane detection and following. Generally, the study of related work is conducted prior to commencing your own research, its aim is to identify the status quo of your relevant field and find gaps that your work could address. However, we have not consulted any related work prior or during the development of our algorithm beyond the documentation provided by the previous team whose algorithm we have been tasked to improve. Therefore, the result of this study is to be used as a bridge to presenting our own work, as well as help us identify potential improvements points. This study, however, does not aim to be an exhaustive review of all the existing lane detection and following techniques found in the field.

Additionally, since most of the literature deals with implementing lane detection in full-sized vehicles, we thought it would be apt to include the lane detection techniques used by Carolo Cup 2016 teams as well. The information is taken from presentation slides that each team prepares for their oral presentations. Unfortunately, since the competition takes place in Germany, all the information presented has been in German as well. As we are not proficient in German, we had to use Google Translate to get the gist of their presentations. Furthermore, due to the nature of oral presentations, little text or in-depth information about their implementations remains, we had to make sense of what little information we had based on our personal experience in developing our own algorithm as well as well observations of their performance in the competition. As the oral presentations are private, we did not have the opportunity to take notes to negate the aforementioned faults. The presentation slides are available by request from the Carolo Cup administration.

The literature for related work has been selected as follows:

1) Search Items: According to Kitchenham and Charters guidelines [16], there is a procedure that consists in several steps in defining the search items:

1) Derive search string from research questions as the most important words.
2) Combine forms of the most significant words derived.
3) Utilize the Boolean OR to alternate between synonyms.
4) Utilize the Boolean AND to combine the most important criteria.
5) Find the key terms.

Therefore, we have concluded that we will be selecting the following terms to find needed material:

((lane) OR (road)) AND ((detection) OR (following))

2) Databases: We have chosen Chalmers Library as our main search engine for the discovery of related work. The library itself often redirects to other libraries. Most of the time, having access to Chalmers Library will grant you access to those foreign sources as well.

Inclusion Criteria	**Exclusion Criteria**
Written in English	Duplicate papers
Published between 2004 and 2016	Does not have relation to research
Machine vision based solutions	Without bibliographic reference
	Does not mention the success rate or successful implementation of proposed techniques
	Patents, News Articles, Books

TABLE II
Study Selection Criteria

3) Study Selection Criteria: Our study selection criteria is presented in Table II.

To lessen author bias and increase the accuracy of the portrayed information, every paper and presentation slides has been investigated and analysed by both authors of this study, therefore the presented results are a product of the consensus of the two.

B. Research Question II

In order to answer RQ2, we undertake the following effort:

- Introduce and explain thoroughly the lane detection and following algorithm implemented by Team MOOSE in the context of Carolo Cup competition of 2015, as an illustrative case study.
- Present our modifications to their algorithm, and how it can improve the stability and precision of their algorithm. The modified version has been used in the context of Carolo Cup competition of 2016 by our team - team Pegasus.
- Showcase the tools OpenDaVINCI [17] provides and how they can aid in the development and improvement of lane detection and following modules.

- Demonstrate and assess the performance of both algorithms with the benchmark tools we have developed. Present the assessment results and describe how we reached said results. The performance study is based on the comparison of two sets of quantitative data, manually derived ground truth data and computer generated data by presented algorithms. The result is presented in graphs.

III. Background

A. Machine Vision

One of the most common ways of detecting lane markings of a previously unexplored road is through vision. Computer vision is achieved by mounting a camera on the car. The camera supplies images, from either photos or videos, that are further processed by the computer. A computer sees an image

Fig. 1. An image matrix and the RGB value of the pointed pixel.

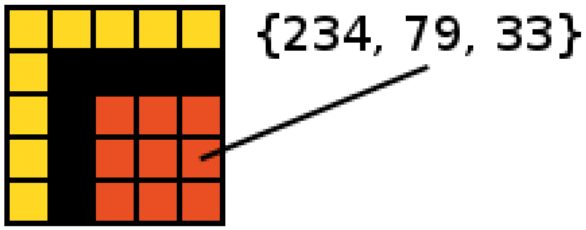
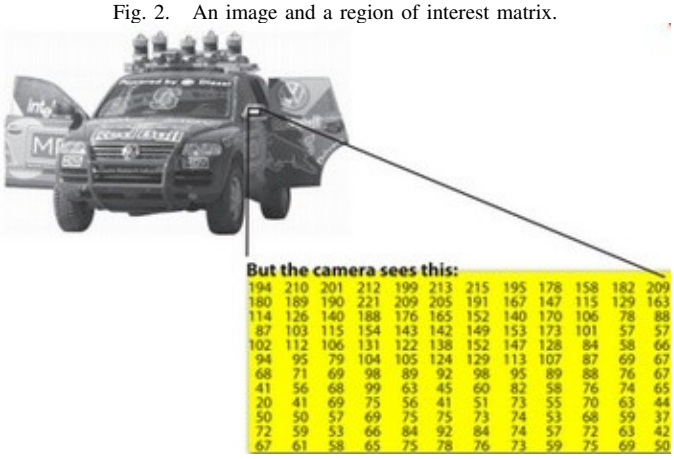


Fig. 2. An image and a region of interest matrix.



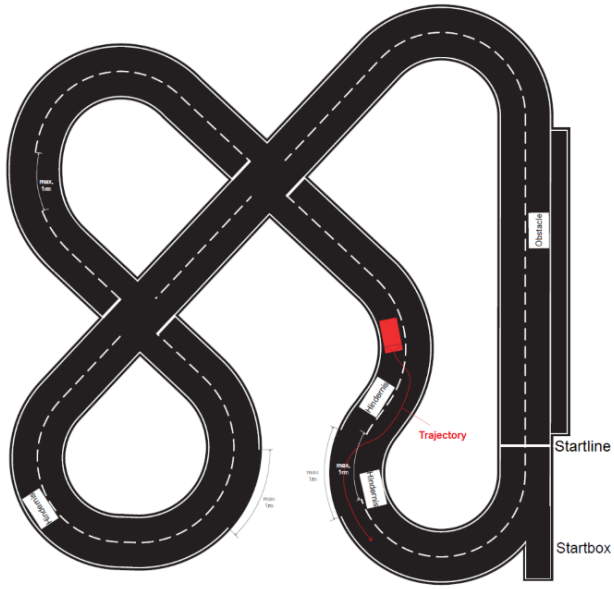
as a matrix of colours [Fig. 1, 2]. Each matrix element holding an array of integers describing the colour using a combination of colour channels, e.g. red, green and blue; cyan, magenta, yellow and black; shades of grey; black and white. An element, unit or item of an image matrix is often referred to as *pixel*.

Unlike humans, who have had millions of years of evolution to become one of the most advanced pattern recognition machines on Earth, a computer does not inherently come with this knowledge. It must be taught to make sense and form patterns from the matrix of pixels it gets.

B. The Carolo Cup Competition

Carolo Cup is a student competition for miniature autonomous vehicles, hosted in Braunschweig, Germany by Braunschweig University of Technology [13]. The goal of the competition is to build a 1/10 scale model of a self-driving vehicle that is able to follow lane markings, overtake obstacles, pass intersections safely and park inside a sideways parking strip autonomously. Points are awarded for the best presentations, the longest distance travelled within a set time with and without obstacles and the fastest sideways parking maneuver. If a car hits an obstacle and/or goes outside the line, the team gets penalised with additional seconds added to their elapsed time. Teams are given two tries to park, in the event of failure after said tries, no points are awarded for the parking discipline.

Fig. 3. Example of a course with broken lane markings and obstacles. [18]



The competition is held indoors, on a scaled down track [Fig. 3]. The track is made of a black coloured rubber base with white tape glued on top denoting the lane markings. It consists of two lanes, with a dashed line in the middle separating the two and solid lines at the ends of each lanes. The track is supposed to emulate a rural environment, contrast to an urban setting where the environment is much more complex (e.g. multiple moving objects, traffic lights, multitude of intersection types etc.).

Each lane is 400mm across while the lane markings are 20mm width, totalling to 820mm horizontally [Fig. 4]. Distance between two dashes is 200mm consistently.

Parts of lane markings are removed on purpose to increase the difficulty of the lane following disciplines. The missing lane markings never exceed over 1 metre length.

At least one 4-way intersection scenario is present. An intersection contains a stop line at its entrance on first encounter from the car's perspective. The stop line is 40mm and spans across the one lane, the lane the car is at currently [Fig.

Fig. 4. Road layout. [18]

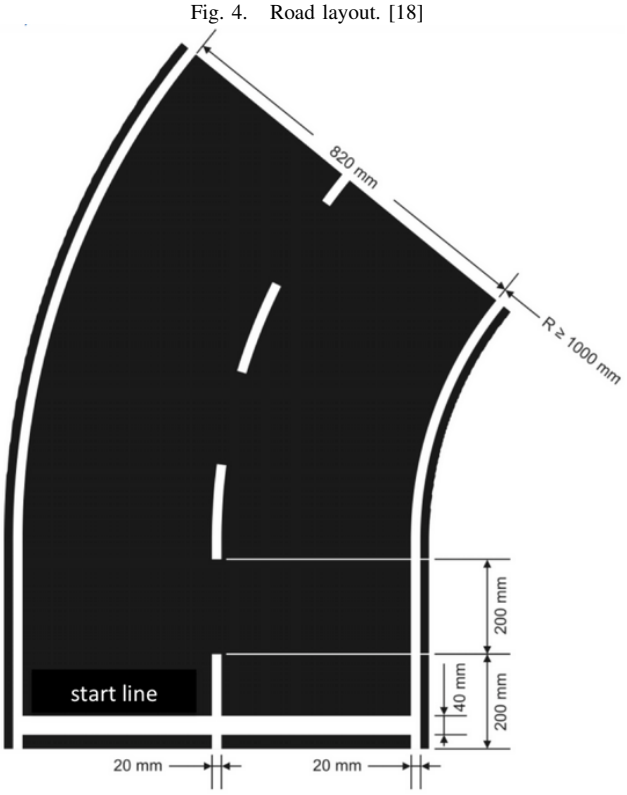
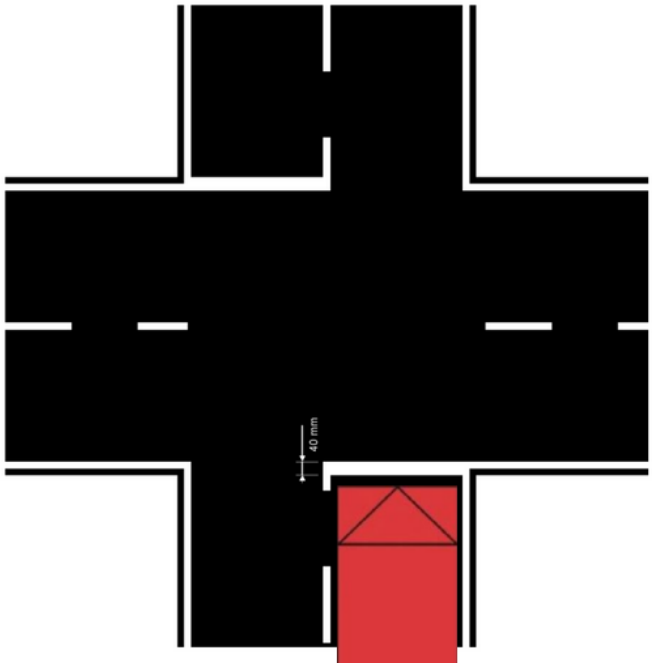


Fig. 5. Intersection example. [18]



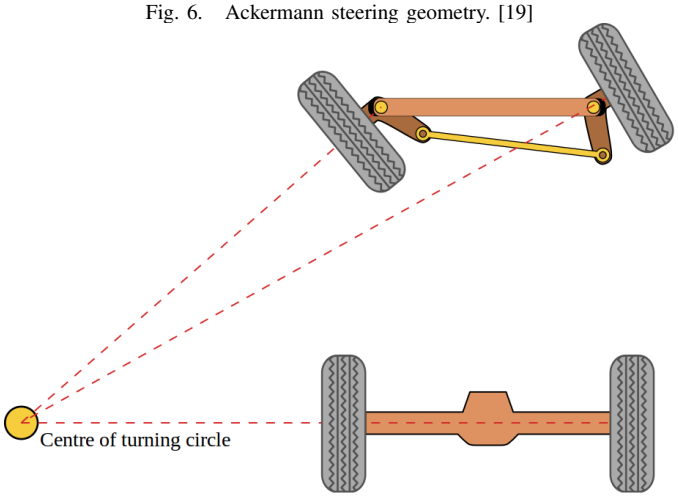
5]. Lane markings are always present during intersections. Intersections do not appear directly after a curve, a straight road segment eases the transition between the two.

The start line denotes the beginning of the track, it spans across both lanes as opposed to intersection stop line which covers one lane only.

Competing cars are supposed to set off from the start box facing the start line.

The lighting conditions indoors are kept at constant 400 lux.

Fig. 6. Ackermann steering geometry. [19]



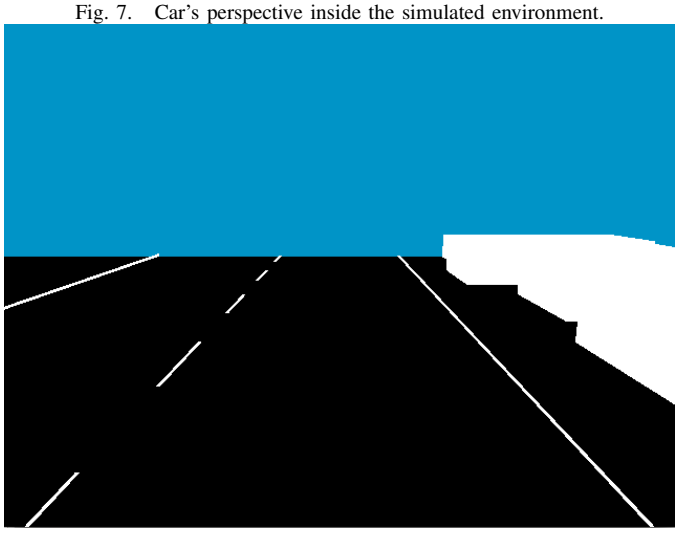
Only cars based on Ackermann steering, or variations of, are allowed to compete [Fig. 6]. Ackermann steering is the standard for commonly met automobiles.

C. OpenDaVINCI and Its Simulation Environment

The OpenDaVINCI framework makes communications between the software components of our car possible by enabling the exchange of data within a local network [17].

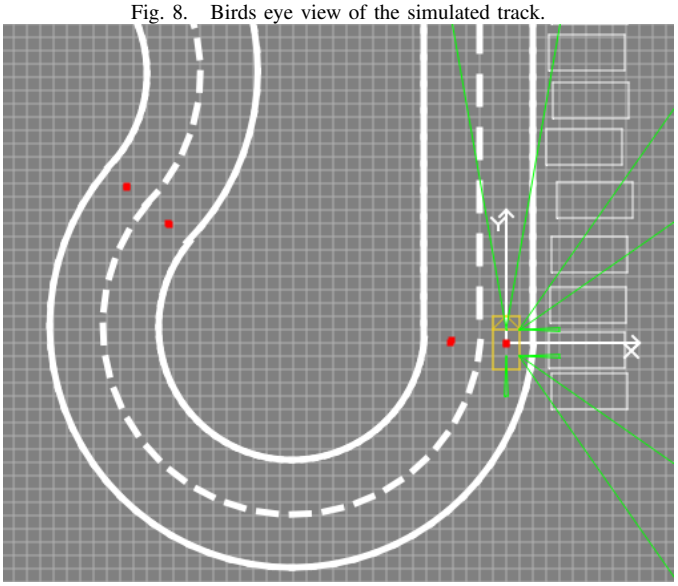
It comes included with a simulation environment. The simulation spawns a car on a track. We are able to get footage

Fig. 7. Car's perspective inside the simulated environment.



of the simulated environment from the car's perspective [Fig. 7] as well as bird's eye view of the map [Fig. 8]. We can also accelerate and steer the vehicle around. The scenario editor lets us modify or create new tracks. With these tools, it is possible to build and test a comprehensive lane detection

Fig. 8. Birds eye view of the simulated track.



and following component from scratch without having the need to own a physical car. Since both the car and the simulation run on the same framework, the codebase and the algorithms remain unchanged, consequently, we can switch between environments without any major difficulty.

Outside simulation, OpenDaVINCI also provides the ability to make video recordings of the camera footage from the car and play the said footage. We have made extensive use of this feature after we have perfected the simulation environment, building tools to both debug and benchmark our lane detection implementation.

IV. Related Work

During the literary research, we have identified a pattern consisting of 4 common steps of successful lane detection and following algorithm implementations: image pre-processing, feature extraction, feature validation and trajectory calculation [20], [21], [22].

1) Image Pre-processing: Images taken directly from the camera often need to undergo certain preparations and transformations to make information extraction easier. These can include the use of one or multiple of the following techniques:

- *Region of Interest:* The image is cut to exclude unwanted information (e.g. visible car, sky, etc) or to focus on a particular area.
- *Greyscale Transform:* Convert coloured pixels into shades of grey.
- *Binary Transform:* Convert coloured pixels into either black or white given an intensity threshold (shade of grey).
- *Blur Filters:* Used to reduce noise and image detail.
- *Inverse Perspective Mapping:* Transforms image's perspective into bird's eye view. This view makes the width of the road equal size at any distance (i.e. in one point perspective view, the width of the road diminishes the further you look).
- *Fisheye Dewarp:* Cameras equipped with fisheye lenses need to be transformed back to normal perspective to simplify distance calculations.

2) Feature Extraction: Information extraction phase. Typically, the intent is to extract features that resemble lane markings, this could be achieved by using the following algorithms:

- *Canny Edge Detection*: As its name implies, the algorithm aims to detect all the edges in an image [23]. It may require threshold tuning to achieve the desired range of edges.
- *Hough Transform*: An algorithm capable of detecting arbitrary shapes [24]. In lane detection, Hough Transform is mainly used to find dominant lines.

3) Feature Validation: Feature validation or fitting is the approximation of extracted features into a smooth path using geometric models. The generated path is later used for various decision making processes typically for trajectory or heading calculations. The aim of these models, usually, is to fit a curve as accurately as possible with given features. The most common ones we have identified are:

- *RANSAC:* A method that requires as much data as available to function as intended, it aims to remove the invalid data by fitting a desired shape to the detected features, i.e. a curve, a straight line, a circle, and then later applies smoothing. RANSAC paradigm is based on three criteria: 1) error tolerance, 2) determine the compatibility with a model and 3) apply the threshold assuming the correct model has been found. [25]
- *Kalman Filter*: A process that filters out noise from given noise data. It is based on a set of mathematical equations that provide a recursive mean to estimate a process and minimizes the mean of squared error, predicting a future state from the previous ones [26].
- *Polynomial Fitting*: Curve fitting is a mathematical technique that is widely used in engineering applications. It consists of fitting a set of points to a curve using Lagrange Interpolation Polynomial. The principal of this method is that given a set of points, the aim is to fit them in a smooth curve that passes through the aforementioned points and the order of the function depends on the number of points. It can be used to approximate complicated curves. However, a high-degree interpolation may also result in a poor prediction of the function between points. [27]

4) Trajectory Calculation: Finally, the vehicle receives the coordinates for desired heading. The trajectory is derived on a 2D plane, it needs to translated to correspond to real world coordinates - 3D plane.

In table III we present our findings of the techniques used in related works. Table IV lists the techniques we have identified that were adopted by the participant teams of Carolo Cup 2016. The information has been gathered from presentation slides that each team prepares for their oral presentations.

Table IV is dominated by references to "Edge Detection", we believe it refers to Canny Edge detection based on resultant

images, it is difficult to infer more as the presentation slides are not descriptive enough.

V. A Lane Detection and Following Algorithm

Each year, teams of students, mainly from German universities, compete in Carolo Cup. Gothenburg University and Chalmers form conjoined computer science and engineering based programmes. The two take a different approach, from the universities in Germany, by creating a new team of students, every year, from both universities' bachelor and master programmes. This has both benefits and drawbacks. It gives opportunity to all students who would be interested to participate and is a potential source for fresh ideas. However, new teams have to either start over or continue the work of the previous teams.

Inheriting code base is not a smooth process due to the nature of competitions. Participants are continuously rushed to implement and test as many features as they can. This leaves little time for in-depth or/and up to date documentation, and maintaining clean and readable code. Consequently, it takes more time and effort to understand the concepts the previous teams have developed.

The competition organisers, at Gothenburg University and Chalmers, try to ease the transition by making students write reports of what they have done at the end of the competition. They also set up workshops introducing their framework and tools, and invite previous team members to give aid in understanding the work they have done. Our team has inherited the code base and a car from the students who competed at the same competition previous year, team MOOSE.

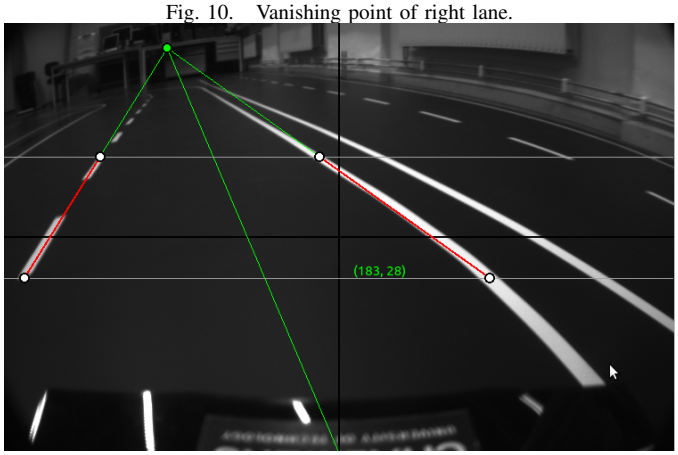
The car we have acquired came with a camera mounted on top at its front. Road markings do not have depth, so a single camera is sufficient. The camera came equipped with a polarising filter and a light sensor. A polarising filter helps minimise road glare, while a light sensor is used to measure the luminosity (lux) of the room the car resides in. The image

Fig. 9. Greyscale image taken from the camera.

we get from the camera is in greyscale [Fig. 9].

As the name lane following implies, our goal is to keep the car, while it is moving forwards, inside the lane it is situated in. Looking down at a lane, its boundaries are always parallel to each other. The car, on the other hand, is directly facing the lane. From its perspective, the lane boundaries are inclined inwards, eventually converging to a point (commonly referred to as a vanishing point) at the horizon line. The horizontal position of the vanishing point denotes the middle of the lane. The car then, has to strive to have its centre line match horizontally the vanishing point to ensure its position in the middle of the lane.

Fig. 10. Vanishing point of right lane.



This is the approach Team MOOSE have decided upon. They have concluded that the vanishing point formed from the convergence of the first dash line and its parallel lane right boundary line segment is good enough to keep the car in the middle in a straight road, while deriving from the first two dash lines guides the car in curved roads [Fig. 10]. Thus it is their aim to detect the first two dash lines, or at least the first one, and the lane's right boundary line to derive the lane's vanishing point. They have set the camera's position at an angle that permits a clear view of the first two dash lines.

Team MOOSE opted to use OpenCV, an open source image processing library. OpenCV provides a myriad of quality of life improvements for manipulating images. It represents images in a matrix data structure of its own and uses the Cartesian coordinates system to navigate around it. The X and Y coordinates originate at top left.

Studying their codebase and reports. We have identified 10 distinct phases to their algorithm within the 4 common steps of lane detection and following we have established earlier in the related work section:

A. Image Pre-processing

1) Extract Region of Interest: The full frame contains the car and the view outside the road. Neither of them present any useful information for lane detection and are unnecessary noise. The smaller an image is, the faster it is to navigate through its entirety, thus it is important to select the smallest region of interest possible while keeping useful information in, i.e. lane markings [Fig. 11].

2) Apply Binary Threshold: There is a high contrast between the road base and lane markings. It is enough to represent the base road as black and lane markings as white

Related Work	Image Pre-processing	Feature Extraction	Feature Validation	Trajectory Calculation
Wang et al. [28]	-	Canny Edge Detection Hough Transform	B-Snake Spline Fitting	Vanishing Point
Jung and Kelber [29]	-	Edge Distribution Function Hough Transform	Linear-parabolic Fitting	General Orientation
Aly [30]	Inverse Perspective Mapping Gaussian Blur	Hough Transform	RANSAC Geometric Fitting	-
Kim [31]	Inverse Perspective Mapping	Machine Learning (Support Vector Machine)	RANSAC Particle Filtering	-
Zhou et al. [32]	-	Canny Edge Detection Hough Transform	Gabor Filter	Vanishing Point
Lin et al. [33]	Region of Interest Gaussian Blur	Sobel Edges	Bayesian Probability Model	-
Mariut et al. [34]	Region of Interest	Hough Transform	-	Mid-lane Line
Ghazali et al. [35]	Region of Interest	Hough Transform	-	-
Li et al. [36]	Region of Interest	Canny Edge Detection Hough Transform	RANSAC Kalman Filter	-
Srivastava et. al [37]	Greyscale Image Binary Image Hybrid Median Filter	Canny Edge Detection Hough Transform	-	-
Borkar et. al [38]	Inverse Perspective Mapping Temporal Blurring Greyscale Image	Hough Transform	RANSAC Kalman Filter	-
Cho et al. [39]	Region of Interest Gaussian Blur	Canny Edge Detection Hough Transform	-	-
Low et al. [40]	Region of Interest Greyscale Image Erosion Dilation Blur	Canny Edge Detection Hough Transform	-	-
Wang [41]	Region of Interest Greyscale Image Binary Image (OTSU) Inverse Perspective Mapping	K-Means Clustering	B-Spline Fitting	-

TABLE III
LANE DETECTION AND FOLLOWING TECHNIQUES FOUND THROUGHOUT RELATED WORK

Fig. 11. Region of interest.

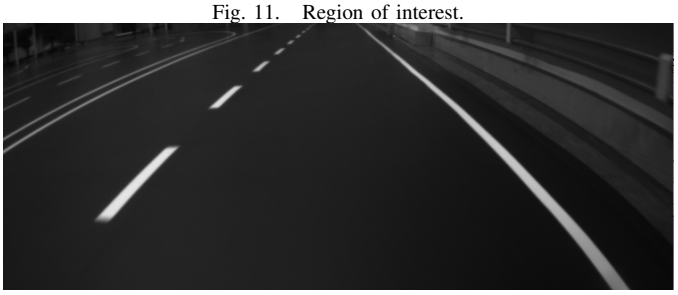
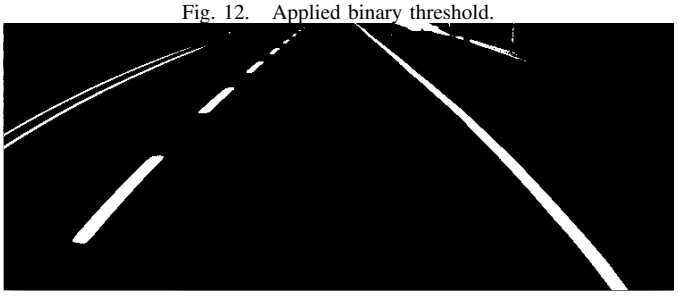


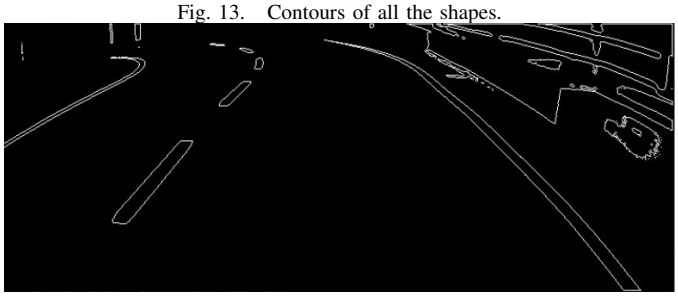
Fig. 12. Applied binary threshold.



to be able to distinguish between the two. It makes it easier to work with two colours instead of a range of shades. OpenCV provides a function to convert the grey pixels to either black or white depending on the threshold value you specify [Fig. 12]. The intensity of a grey colour is expressed as a value between 0 (black) and 255 (white), giving us 253 shades of grey. Any grey intensity below the threshold becomes black, any higher or equal - white. This step also removes a lot of the noise, such as dust specks or dirt as they are seldom brighter than the lane markings. The threshold value is dynamically defined from the room's ambient light measured by the light sensor.

B. Road Features Extraction

Fig. 13. Contours of all the shapes.



1) Find Contours: With the help of OpenCV, it is possible to get the contours of all the shapes in the frame [42] [Fig.

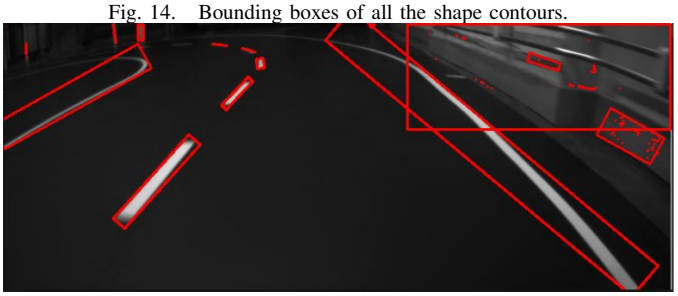
Team	Image Pre-processing	Feature Extraction	Feature Validation	Trajectory Calculation
Berlin United	Greyscale Image Region of Interest Inverse Perspective Mapping	Edge Detection	RANSAC	Target Point
CDLC	Greyscale Image Binary Image Inverse Perspective Mapping	Points Detection	RANSAC Polynomial Fitting	Target Point
eWolf	Greyscale Image	Edge Detection	?	?
GalaXIs	Greyscale Image Fisheye Dewarp Inverse Perspective Mapping	Edge Direction	Polynomial Fitting	Mid-lane Line
ISF Löwen	Greyscale Image Fisheye Dewarp Inverse Perspective Mapping	Edge Detection	Geometric Fitting	Target Point
it:movES	Greyscale Image Region of Interest	Edge Detection	?	?
KITCar	Greyscale Image Binary Image	Scanlines	Polynomial Fitting	Mid-lane Line
NaN	Greyscale Image Binary Image (OTSU)	?	Polynomial Fitting	Target Point
Ostfalia	Greyscale Image Contrast Brightness Region of Interest	Edge Detection	Geometric Fitting	Target Point
Pegasus	Greyscale Image Region of Interest Binary Threshold	Scanlines	Geometric Fitting	Vanishing Point
TUM Phoenix	Greyscale Image Inverse Perspective Mapping	Line Segments	Kalman Filter Geometric Fitting	Target Point

TABLE IV
LANE DETECTION AND FOLLOWING TECHNIQUES USED BY CAROLO CUP 2016 TEAMS

13]. The shapes are separated from each other by black space.

2) Approximate Contours: OpenCV provides yet another useful function that approximates the number of points in a contour given distance between two points [43]. The aim is to find the general direction of a shape at its upper segment, therefore, all the contours are approximated to 4 points forming a rectangular shape. The dramatic reduction of points in each contour makes the next processing steps less time consuming as they have to process 4 points each contour now instead of potentially hundreds of points.

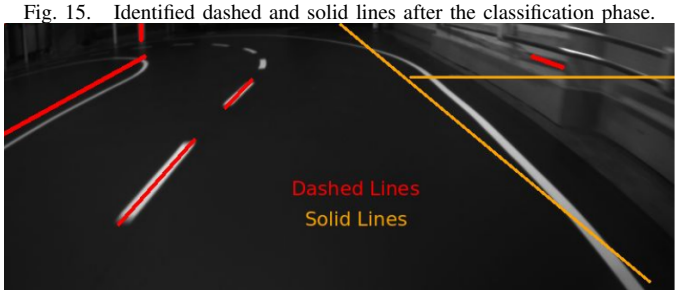
Fig. 14. Bounding boxes of all the shape contours.



3) Get Rectangles: Each approximated contour is then transformed into an OpenCV Rotated Rectangle [Fig. 14]. This data structure helps us find out the angle each contour is rotated at.

C. Road Features Validation and Fitting

Fig. 15. Identified dashed and solid lines after the classification phase.



1) Classify Lines: The rectangles are approximated into lines according to their vertices and are then classified into solid or dashed lines. The classification is done by comparing the ratio between width and height of the rectangles or their

length compared to the longest rectangle [Fig. 15]. Sometimes, solid lines may be miss-classified as dashed lines, this step of the algorithm is ran again during later phases to minimise this risk.

2) Set Road State: During the classification phase, the road state is set to either normal mode, intersection mode or start box mode.

Intersection mode is triggered when one of the rectangles lays flat 90° and occupies a big portion of the image.

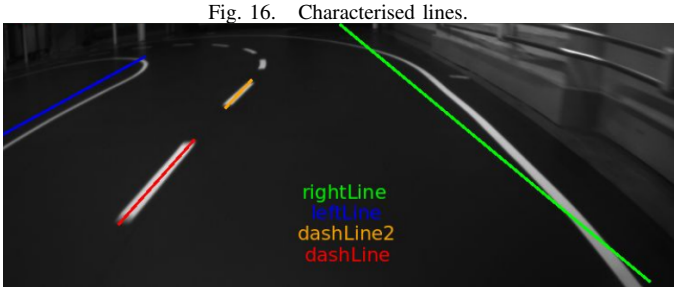
While the start box mode is triggered when there are two rectangles occupying a considerable amount of the screen while laying inwards to each other. Otherwise, the road state is set to normal.

3) Filter Lines: Lines too far to the left or to the right side of the screen, outside the road boundary lines, are removed. The lines that are too far ahead relative to other existing lines are filtered out as well as the car uses the lines closer to its position to guide itself.

4) Characterise Lines: This step categorises the remaining lines into left boundary line, right boundary line and tries to form a dashed lines curve.

The dash lines are sorted by their Y axis position, from bottom to top. The dash curve is considered to be found when it contains two dashes. If there is only one dashed line discovered during the classification phase, it is immediately selected as the dash curve. At this point, it is not certain whether the dashed curve formed is the right one, the algorithm is ran again for the remaining dashes to create multiple dash curves. To pick the right one, the generated dash curves are compared to previously selected curve from earlier frames.

Fig. 16. Characterised lines.



The boundary lines are identified from the remaining solid lines. Left and right lane markings are assumed to lean inwards. The left line is assumed to have an angle of more than 90° relative to the base of the frame, while the right line less than 90°. Their position relative to the centre of the frame is also considered, i.e. the right line cannot be to the left of the centre line. The identified lines are than compared to the lines from the last frame, the difference in their absolute positions should be minimal [Fig. 16].

D. Trajectory Calculation

Finally, the vanishing point is derived from the supposed eventual intersection of the dash curve and the right line. The vanishing point of the left lane is also derived, it is used when the car needs to switch lanes to avoid obstacles. No vanishing point is created during intersections as there is too much interference.

VI. Benchmarking the Algorithm

Evaluating the performance of lane detection and following remains challenging to this day. A standard performance measurement process has not been established yet. Most researchers report the success of their lane detection methods qualitatively, others report hours of successful driving. However, without comparing the algorithm results to manually derived standard, there is no way to quantifiably measure and prove the results' correctness [21][44]. Few researchers resort to using ground truth data due the absence of public data sets [21]. Ground truth data generation is an arduous and time consuming task.

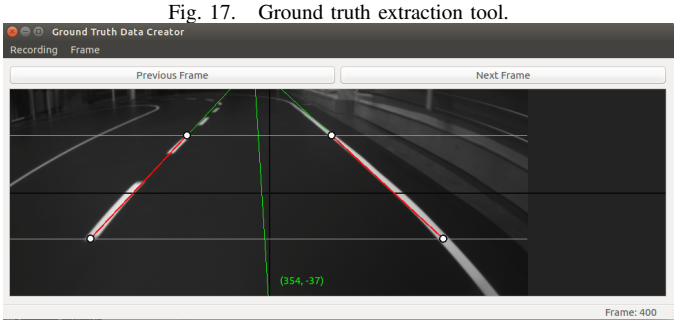
Team MOOSE have spent substantial time and effort manually plotting vanishing points to each frame of the video recordings they have made. They have taken a recording of the car (radio controlled) completing the track once. Additionally, they have split the recording into specific parts denoting the following scenarios: straight road, left curve, right curve, S curve, road nearby, start box, intersection, dashed line missing, right line missing. This makes it easier to localise errors.

Unfortunately, we have discovered their data to be flawed. They have derived the vanishing point using the first dash line, both theirs and our algorithm depend on the second dash line for the correct trajectory. At the times when the right boundary line was not visible in the frame due to the car's acute position, they were plotting the right line at the side of the screen instead of estimating the line outside the frame.

In order to correctly measure the performance of both algorithms, we were bound to recreate the ground truth data. We have developed a ground truth data creator tool to ease the process.

A. Ground Truth Data Generation

Fig. 17. Ground truth extraction tool.



The ground truth data creator tool is able to load recordings and traverse them both forwards and backwards while displaying each frame in the main window. Its main purpose is to let us draw the lines that form the vanishing point in a visual and user friendly way [Fig. 17].

The moment we plot the last point, it derives the vanishing point automatically and draws it on the screen. If we are

unsatisfied with the result, we get the option to delete the plotted points in the current frame and try again.

Sometimes, the car stops for one reason or another (e.g. during intersections), for the next hundreds of frames, the lane markings would stay the same so the vanishing point would be identical as well. Instead of having to plot same lines over and over again, we added the feature to copy the lines of the previous frame. This way, we can skip the moments when the car is stationary in seconds.

The main window contains extra space to the right side of the frame, this allows us to guesstimate the location of lines that are off-screen.

Another useful feature is being able to jump to any frame number within the recording.

Once we are done plotting the vanishing points, the program exports, into a CSV file, the coordinates of the vanishing points and the line segments it derived from.

While we are unaware how Team MOOSE created their ground truth data, they claim it took them at least 2 weeks to do so. With our tool, we are able to recreate the same data set in under a day.

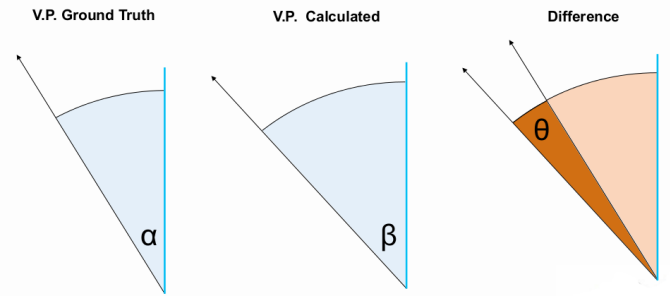
B. Visualising the Data and Results

To help us visualise this data, we have built a tool that spawns the OpenDaVINCI player and plays each recording while exporting the calculated vanishing points, derived by their algorithm, for every frame. The resulting CSV files are then compared with the ground truth files.

Our final goal is to horizontally align the car's centre line with the vanishing point, at that moment, the car would be perfectly positioned in the middle of the lane it is in. It would be useful to know then, how off course the centre of the car is relative to the vanishing point. To accomplish that, we extend a line from the vanishing point to the base of the centre line of the car (centre of the frame). The angle formed between the two is the error angle we are looking for.

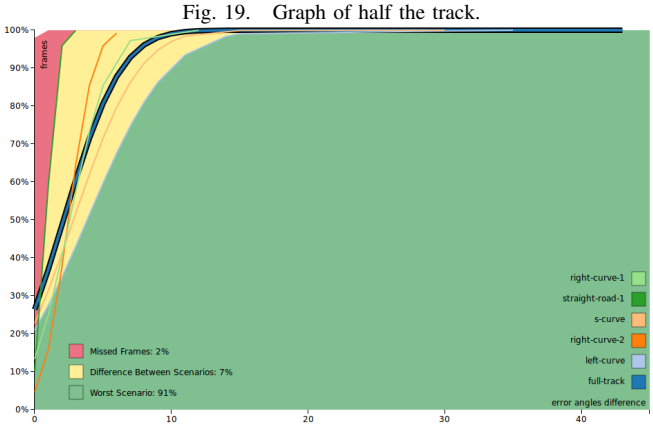
Thus, to regard a calculated vanishing point as correctly identified, its error angle should match the error angle of the ground truth data. The ground truth data itself is, however, not perfect. As its drawn by hand, the points vary from frame to

Fig. 18. Error Angles Difference.



frame, it would be more appropriate to consider that the closer the difference between calculated and ground truth error angle is to 0, the more accurate the point is [Fig. 18].

Fig. 19. Graph of half the track.



All things considered, assessing the performance of the road as a whole is not as useful in pinpointing problematic areas. Thus, along with benchmarking the whole track, we cut the road further into logical sections representing their shape and scenarios, e.g. straight road, left curve, right curve, intersection, and plot them on the same graph. For the sake of demonstration, we have created a graph of half the road to make it easier reading the data [Fig. 19].

Each scenario is plotted on the graph as a cumulative distribution function (of normal distribution), this allows for ease of assessment at a glance. Taking the left curve scenario as an example, we can determine that the scenario has about 89% chance of each frame being less or equal 10 degrees error angle difference. Going further to the left, we find that the scenario has about 55% of having error angle less than or equal to 5 and about 22% to have the error angle of 0 (the perfect case). This can be repeated for each scenario. The full track is always to be found somewhere in the middle of the rest of the scenarios as the rest are derived from it. Thus, to judge problematic areas we look at all the scenarios represented to the right of the full track curve, they are the ones bringing the average down.

The next step in address the scenarios would be, replaying the recordings at the problematic areas, trying to determine the cause for the inaccuracy, modifying the algorithm in response and then rerunning the benchmark tool. If you are lucky, you would have fixed the problem without badly affecting the rest of the scenarios. While debugging, we display the results of all the algorithm steps for each frame, thus having an in depth understanding of when things went wrong in the process of deriving the vanishing point.

To make it even easier to judge the situation at a glance, we have divided the graph into 3 areas. The right-most integral (green) represents the worst scenario, the middle integral (yellow): difference between the best and worst scenarios and the left-most integral (red): missing frames.

The worst case scenario is calculated by taking the mean of all the error angles in a scenario, the scenario with the biggest mean is the worst case scenario and the scenario with the smallest mean is the best case scenario respectively. The

remainder is defined as missed frames, an area left to be filled.

Our aim is to move the right-most integral towards the left as much as possible, improving the worst scenario. The secondary aim is to lower the maximum difference error angle, although it's not as important, as most of the times, the high error angle frames happen in spikes so they normally do not affect the car's movement in any significant way.

The perfect scenario would be to have the maximum error angle be 0 and all the frames be at 100% probability of having angle 0, basically forming a vertical line.

The colours have been chosen deliberately to invoke positive or negative attitude towards the presented data, green signifying good, yellow: warning and red: undesirable.

In active development of the benchmark tool, we have cross-checked with Matlab to ensure the correctness of our graphs. The reason Matlab has not been used as the main source is, we needed a one button solution for performance analysis to make the process as streamlined as possible, without the need to own particular software or knowledge in operating said software.

During the improvement process of our algorithm, we used the graphs as a quantitative way of proving that our changes are beneficial. Figure 20 presents the final results of our new algorithm and team MOOSE's algorithm, and compares the two.

To be able to compare two builds accurately, both graphs must have the same X and Y-axis scales. Consequently, for the X-axis, we have chosen the maximum angle of 45, even though there is a possibility of one of the graphs having a lower maximum error angle. The angle 45 represents the maximum angle the wheels of our car can turn. We do not necessarily have to know the exact maximum error angle as the trend of the worst scenario curve should give us an idea of how bad it is.

Having Git as our source control, we conducted all our development work in a branch. Only positive changes would then be merged with the master branch. Table V summarises the improvements.

VII. Our Improved Lane Detection Algorithm

While the new teams are allowed to start from scratch if they so desire, we are strongly encouraged to improve previous team's work instead. This particular lane detection and following algorithm is at its third iteration of improvements.

The car we've inherited has been revamped extensively [Fig. 21]. The camera angle has not been changed, however, as the vehicle's control loop has been tuned to that particular camera position.

The assumption that the car is always inside its lane drives the whole algorithm. There were no efforts made on trying to return the car back to the lane in case of erroneous prediction. During the competition, we are allowed to use a remote control to recover our cars, it is both faster and safer than any automatic implementation.

The algorithm has been reduced to 7 distinct phases within the 4 common steps identified earlier in the related work section:

A. Image Pre-processing

1) Extract Region of Interest: This phase remains the same as before due to the aforementioned vehicle control loop tuning.

2) Apply Binary Threshold: Instead of relying on the light sensor to judge the right threshold value, we decided to do it manually ourselves. While test running the vehicle on our track, we noticed the light sensors would report irregular luminosity spikes which made the car respond unexpectedly. During the competition, the light in the room is at constant 400 lux. Therefore, we manually apply the threshold value by making a short recording and then fiddling with the threshold value until we remove any road glare while keeping as much road information as we can.

B. Road Features Extraction

1) Extract Road: Instead of letting OpenCV loop through the whole image to find all the objects, we opted to go with a more surgical approach. We aim to find the first two dash lines, the right road line and the left road line and save their contours immediately while scanning the image only once. The moment a particular line is found, the algorithm should stop searching the area.

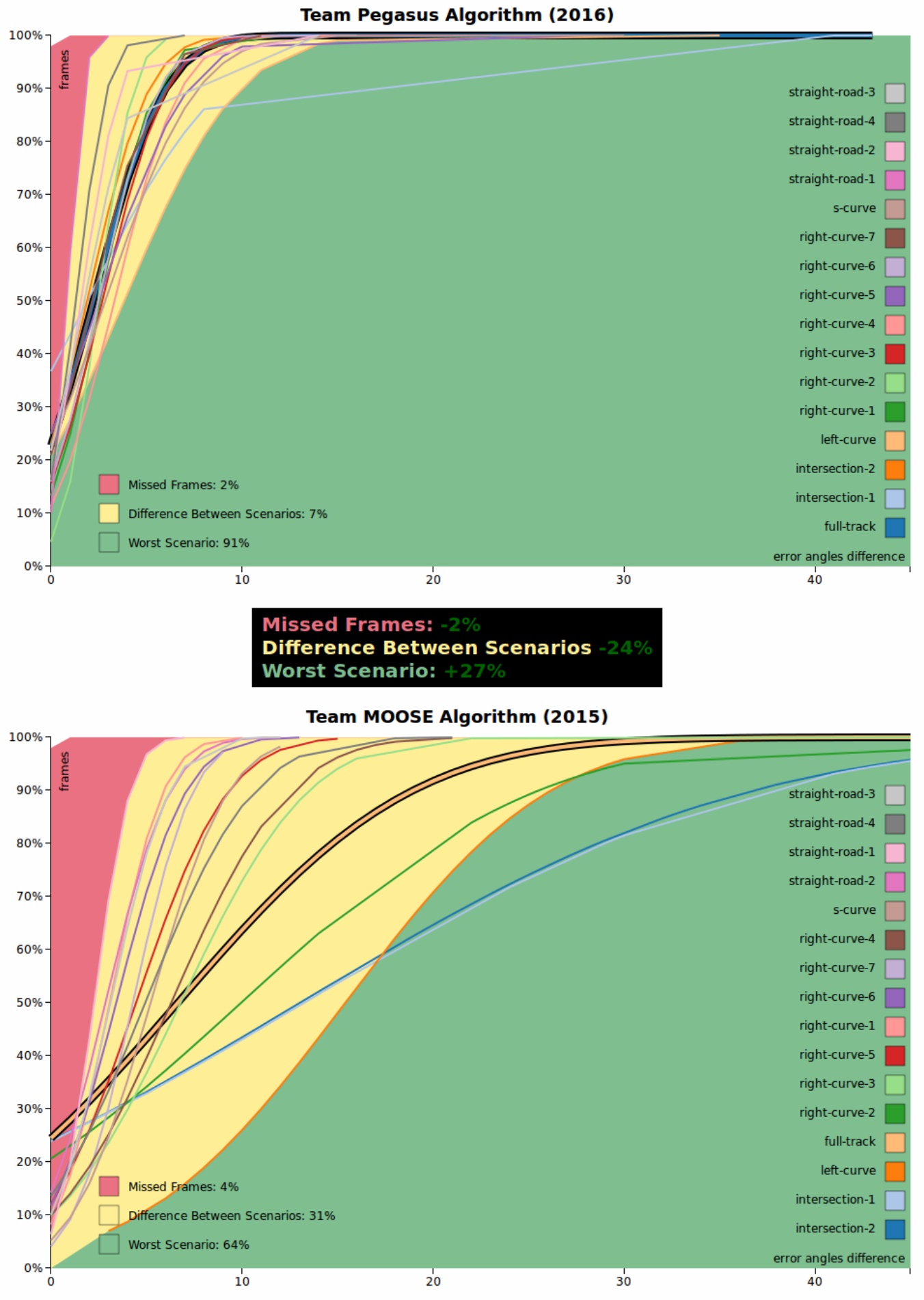
Finding the right line:

1) The algorithm starts scanning from the centre of the frame towards the right side, from the bottom of the image towards the top. It is searching for a white pixel.
2) The moment it hits a white pixel, it assumes it has found the right line so it notes the pixel's position as part of the right line's contour.
3) When it encounters a white pixel for the first time in the row it is scanning in, it notes down the position for future use, we shall refer to it as point *FWP*.
4) It continues scanning towards the right until it hits a black pixel and notes the pixel position (-1 pixel) as part of the contour again.
5) While traversing the white pixels, it keeps the position of the last found white pixel for future use, we shall refer to it as *LWP*.
6) After it hits the black pixel or it meets the other end of the frame, the current row stops being scanned. Sometimes, objects stand right on top of a lane marking, in that case, the algorithm would continue scanning until it reaches the end of said object as well. To fix this problem, we can set a maximum number of white points to scan per row for a particular line, we shall refer to it as *MaxWP*. If the algorithm exceeds *MaxWP*, it should stop scanning and move on to the next row.

The search starts again, but in the row situated above, from the X-coordinate of point *FWP*, with an offset towards the left. The offset is necessary to be able to accommodate the line curving towards the left. Lane markings have predictable shapes, in our experience a 3 pixel offset is enough.

Steps 2-6 are repeated until the line ends at the top or the algorithm reaches the end of the frame at the top. The

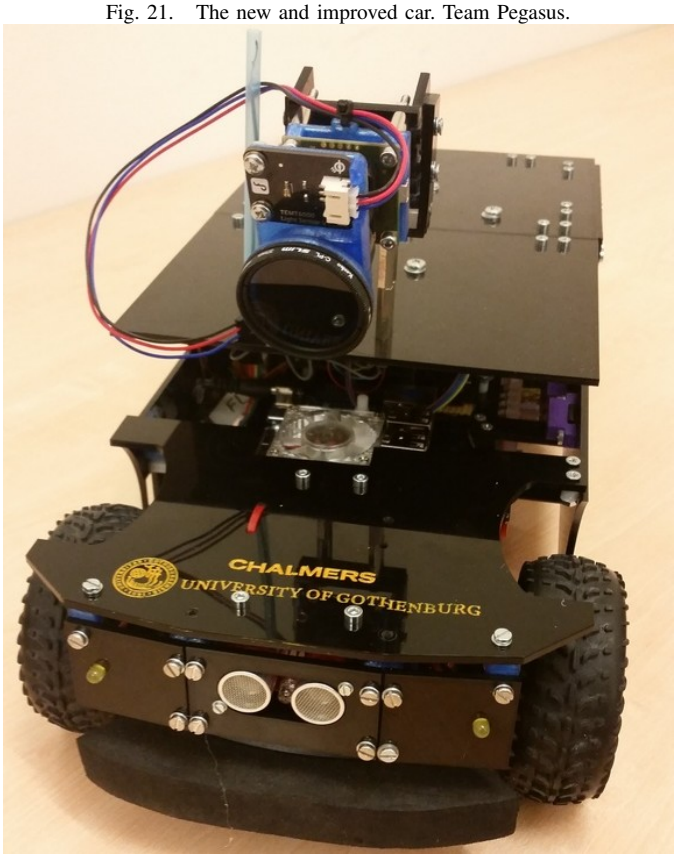
Fig. 20. Difference between Team Pegasus and Team MOOSE algorithm.



Area	MOOSE 2015	Pegasus 2016	Difference	-
Missing Frames	4%	2%	-2%	Less is Better
Difference Between Scenarios	31%	7%	-24%	Less is Better
Worst Scenario	64%	91%	+27%	More is Better
Frames Per Second	25-35 FPS	90-100 FPS	+65 FPS (3x faster)	More is Better

TABLE V
PERFOMANCE AND PRECISION COMPARISON BETWEEN TEAM MOOSE AND TEAM PEGASUS ALGORITHM

Fig. 21. The new and improved car. Team Pegasus.



algorithms knows when it reached the top of a line if it arrives, without having met any white pixels, at X-coordinate of point *LWP* + an offset. This offset could be *MaxWP*.

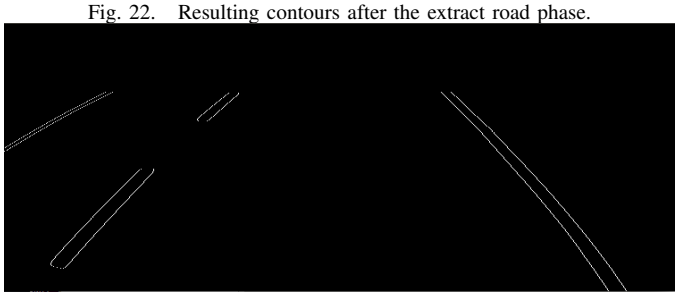
To make sure that the detected line is not actually dust, dirt or any other type of similar noise, the algorithm counts the number of points in the contour. If there are less points than the specified threshold, the contour is discarded and it continues scanning from where it left off the last time. This kind of noise can be reduced by either increasing the binary threshold value or/and eroding the image by set number of pixel.

For finding the left line, the algorithm follows the same steps as in finding the right line except it starts scanning at the left middle side of the frame (X: 0, Y: Frame Height / 2) and it scans towards the left with and offset towards the right, and obviously saving relevant white points to the left line's contour. The camera is positioned in a such a way that the left line is only visible in, assuming we cut the frame in 4 equal sectors, the top left sector of the image.

Finally, to locate the dash lines, the algorithm again follows the steps for right line except it scans towards the left and mirrors any other relevant offset directions. Once it finds the top of the first dash line, it continues its search for the second dash line.

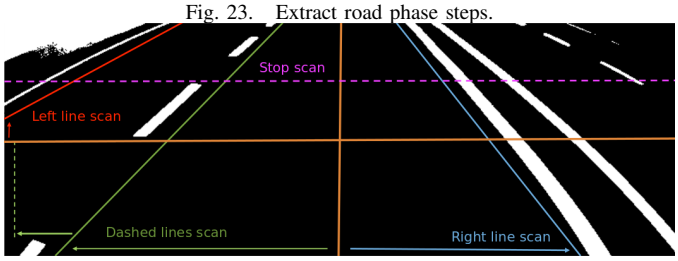
The search for all the lines happen in parallel in each row with the following priority: right line, left line, dashed lines. The reason dashed lines are being detected last is, the algorithm keeps the last points of both right and left lines at that particular row to not let the dashed lines scanner come close to either left or right lines to prevent solid lines be detected as dashed lines. When the end of the second dash line is found, the main loop quits so scanning of the side lines stops at that point as well.

Fig. 22. Resulting contours after the extract road phase.



The resulting extracted contours can be seen in Figure 22.

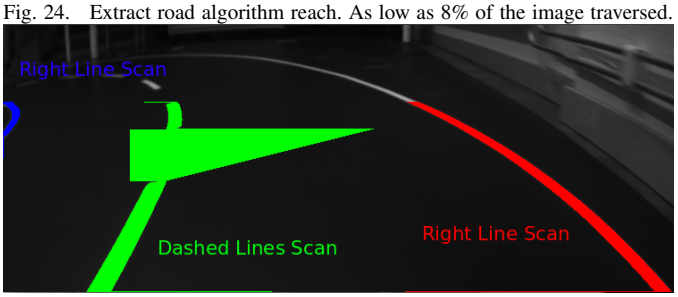
Fig. 23. Extract road phase steps.



An overview of the extract road steps is shown in [Fig. 23]

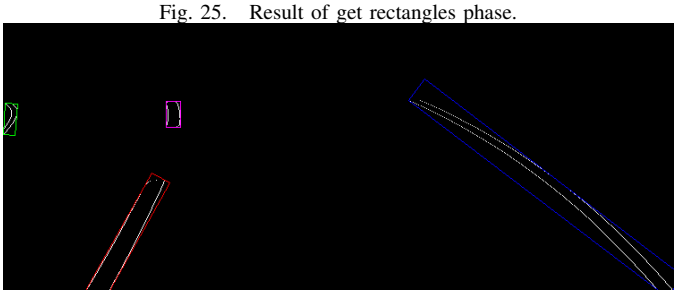
We use pointers to loop through and access the contents of a matrix. Looping through a matrix using a pointer is the fastest way, OpenCV's at() function is designed for random access [45]. The reach of the extract road algorithm of an

Fig. 24. Extract road algorithm reach. As low as 8% of the image traversed.



average frame is demonstrated in Figure 24. As low as 8% of the image traversed, much more efficient than looping through the whole image with OpenCV's find contours function.

Fig. 25. Result of get rectangles phase.

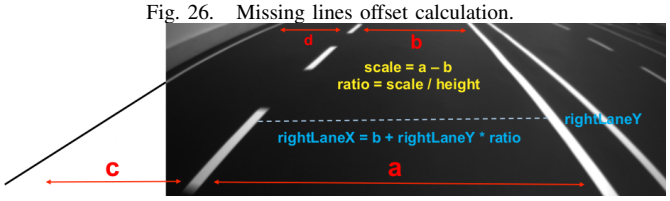


2) Get Rectangles: Remains the same except there are at most only 4 contours to process. Because there are so few contours, approximating their points does not yield any performance increase [Fig. 25].

C. Road Features Validation and Fitting

1) Extract and Filter Lines: Lines are approximated from rectangles and then filtered out by their size and angles. Same lane markings exclusion criteria apply as in the previous algorithm.

Fig. 26. Missing lines offset calculation.

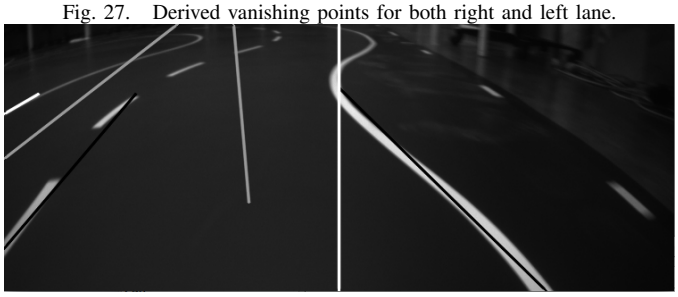


2) Draw Missing Lines: Missing lines are derived by mirroring existing lane markings. Fig. 26 describes the calculations. The algorithm assumes that the car is perfectly aligned on the Z-axis facing the lane. However, this is rarely the case, especially when the car turns during curves. Still, it provides a good enough approximation for the car to stay within the lane until it regains sight of the next lane markings. The competition rules do not require for the vehicles to be perfectly centred within a lane, merely for them to stay in the lane.

D. Trajectory Calculation

Remains the same as before [Fig. 27].

Fig. 27. Derived vanishing points for both right and left lane.



VIII. Analysis and Discussion

A. Results Discussion

During the related work study, we have noticed many similarities to our implementation, specifically the pre-processing stage. Both the Carolo Cup teams and the bibliographic sources tend to use complex curve fitting models to derive their lane markings. We did not have such need as our only concern is to get the trend or the direction of the intersecting lane markings, the benefit of not transforming the image to a bird's eye view. As for features extraction phase, both the bibliographic sources and most of the Carolo Cup teams follow the same approach. We have developed our own custom process, more fit for the problem of detecting lane marking for the purpose of lane following as it scans from the position of the car outwards instead of using a more generalised approach by scanning the whole image, from top to bottom, as we proved to be wasteful and less precise.

As the graphs indicate, our improvements have had positive effects over precision and stability of the inherited algorithm. There are less missed frames, less difference between scenarios and the worst case scenario is more stable.

The increase in precision is attributed to the extract road phase. Compared to the previous algorithm, ours does not have to guess from the multitude of shapes which of the lane markings they belong to anymore. It detects at most 4 lines. Since we stop scanning once we find our lane markings, the trajectory calculation is unaffected by objects outside the road. This has been a great boon during the parking sequence as it has not been affected by the parked cars along the road, as well as in the instances when there is another road nearby. Missing lines recovery is also more reliable, even though it does not account for Z-axis rotation.

Another big improvement is the processing speed. The processing speed increase has been a side effect of our precision improvement efforts. Last year's algorithm ran at an average 35 FPS, while ours runs at round 100 FPS, effectively tripling the speed. This increase in performance has had great effects on the car's lane following ability as it can process more frames at a time thus make decisions much faster and have a more up to date vanishing points. We were able to run the car at greater speeds than possible before.

However, some benefits of the previous algorithm have been lost. One of the more advanced disciplines in the competition is lane following with obstacles on the road. The objective is

the same as in regular lane following, except the objects have to be avoided. In preparation for this task, Team MOOSE have added the ability to derive the vanishing point of the other lane, so theoretically, you simply would need to follow the other vanishing point instead. This feature is still present, however, our algorithm only works on the right lane at the moment as we have not planned on undertaking the overtaking discipline. In order to make the algorithm work on the left lane, we simply flip the scanning direction of each involved algorithm step.

Another issue arises while trying to switch lanes. Our algorithm has no inherent ability to detect on which lane it is, so it would not know at which point should the scanning direction of extract road phase flip. Adding such a feature could possibly undo the performance benefits of the new algorithm if it involves scanning the whole frame. More lighter solutions would be welcome such as using scan lines at great intervals, but this method tends to be prone to noise or busy background outside the road. The most straightforward solution is to simply execute a blind maneuver and time it so it at least crosses the dashed lines before it resumes to detecting lane marking again.

One more downside is, if there are objects or neighbouring lanes near a missing lane area, the algorithm will treat them as a lane marking affecting the correctness of our trajectory. However, during the competition, we have not met such cases or they have not affected the stability of the lane following significantly.

Previously, teams had to rely on observations of the car running on the track to determine the effects their changes had. The benchmark tool greatly increased our efficiency and confidence in making changes by letting us see what areas have been affected. If our changes did not lead to improvements, the tool could at least confirm that no unexpected or negative effects occurred.

B. Threats to Validity

A study validity is the level of trustworthiness of the presented results [46]. We have deployed the threats of validity model by categorising them in the following categories:

1) Internal Validity: Internal validity is defined as factors and risks that are unknown to the person conducting the research and can affect the researched factor [46].

In a literature review, there is always risk of author bias. This is especially true in research that is based on summarising and getting an understanding of the concept relayed in a paper as it is reliant on author's technical expertise and good will to remain neutral. This paper is not an exception. We have tried to minimise author bias by making both of us analyse each paper and reach a consensus on the results, even so, we are still bound by our technical knowledge as we are new comers to the field.

Another problem in our related work study is the presentation of the techniques used by Carolo Cup teams of 2016. The techniques were gathered analysing the slides they have prepared for their oral presentations. As the teams are all German, the slides were written in German as well. We do not posses knowledge of the German language so we had to use Google Translate, the translations could be inaccurate. Furthermore, the presentation slides were made for an oral presentation which contain little information by design as the speaker ought to relate most of it. We had to infer the information we presented from keywords and images of the results they have included.

2) External Validity: The lane detection algorithm we have implemented has been created in a controlled environment within the rules and the setting of the Carolo Cup competition. While it may serve well in use in other similar competitions, it is not directly immediately suitable in a full sized vehicle on real roads as it does not handle different light and weather conditions and complicated road layouts. However, it can be useful as a base, a boilerplate for any efforts to implement lane detection in full sized cars. The benchmark tool, on the other hand, bears no such restrictions. It can be used to its full potential and benefit from all its features benchmarking the lane detection of a full sized car on a real road by following the same methodology we had in our miniature vehicle.

3) Construct Validity: Construct validity knowing whether the study has been done the way the researcher planned and thought of to correctly answer the research questions [46].

We have not found a similar benchmark tool in quantitative way able to compare between two builds and aim to increase the accuracy and precision with visual results that have been implemented before. We had to implement it ourselves. We only found one study that highlights its importance. Additionally, we have manually entered the data and checked its correctness with Matlab to make sure the output graphs by the benchmark tool coincide with Matlab.

IX. Conclusion

This research, consists of a literature review on contemporary tactics to implement the lane detection and lane following functions in autonomous vehicles, as well as an excerpt of the related work that was conducted during our participation in an international competitions on autonomous vehicles in Braunschweig, Germany. The bibliographic research provides an overview of the most common tactics found in lane detection implementations.

Particularly, we have outlined the findings from fourteen published papers and have defined four phases, typically involved in lane detection and following, which include image pre-processing, feature extraction, feature validation and trajectory calculation. Our classification of the findings, makes apparent the most common approaches for each stage, allowing us to acquire a synopsis of the state of the art work in the field.

Next, we have extensively outlined our practical involvement with the investigated functions, which were realized on a miniature autonomous vehicle. Specifically, the prior approach on the matter, was drastically improved, translating into higher software quality and performance. What is more, this was documented and quantified by a graphical evaluation tool that we developed. The specific tool, enabled the visual

validation of the various changes that were committed, providing benchmark oriented guidelines to the developers, on whether the code should be deployed or not. Other important improvements to the lane detection and following approach, that aimed to increase the performance, included a feature to extract the "road" from an image frame. This accelerated the volume of necessary calculations and dramatically decreased the processing time.

However, our algorithm is not necessarily confined to miniature vehicles. Even though it does not handle different light and weather conditions, it can still be used to kick-start a lane detection and following project from scratch even for a full sized vehicle on real streets. Full sized vehicles can still take full advantage of the benchmark tool we have developed and benefit from all its features, i.e. quantifiably measure the performance of the lane detection algorithm, by following the same methods we applied to our miniature vehicle.

For the future, there are a lot of suggestions, that would lead to the improvement of the lane detection and following features. PID controller tuning, would allow for smoother transitions. Additionally, obstacle avoidance could be incorporated to the current algorithm, therefore, increasing the product's functionality. Finally, a method should be devised in order to programmatically differentiate between the two scenarios, of the stop and start line, that are particularly valuable in the context of the autonomous vehicles competition in Germany.

X. Acknowledgements

We would like to extend our sincerest gratitude to Dr. Christian Berger for his support and guidance throughout his course, the Carolo Cup competition and this study, and for introducing us to the intricacies of the automotive industry. Special thanks to our Carolo Cup supervisors Martin Holder and Thomas Petig for having the patience getting our benchmark tool right and guiding us with our algorithm improvement efforts. Furthermore, we would like to thank Team MOOSE for taking their time in aiding us early in the development of the new algorithm by making sure we understand their previous implementation and giving pointers for possible improvements. We would like to thank Bosch for hosting a wonderful event, and giving us an opportunity to test our car and our lane detection and following algorithm on a foreign track, it was a good wake up call to double up our efforts in improving the algorithm. Special thanks to Chalmers, HiQ and Delphi Automotive for sponsoring the equipment we needed to build the car and for our travel and stay in Germany during the Carolo Cup competition event. We would also like to thank Dr. Jan-Philipp Steghöfer for pointing out the faults of this study and helping us address them. And Imed Hammouda for his support throughout our studies in our programme. Last but not least, we would like to thank our colleagues within team Pegasus for the great work they have done and the fun we have had during the Carolo Cup competition of 2016.

Thank you, everyone.

References

[1] IHS Automotive. Emerging technologies: Autonomous cars-not if, but when. Technical report, IHS Automotive, 2014.
[2] Navigant Research. Advanced driver assistance systems and the evolution of self-driving functionality: Global market analysis and forecasts. Technical report, Navigant Research, 2015.
[3] Andreas Knapp Arne Bartels, Ulrich Eberle. System classification and glossary on automated driving. Technical report, AdaptIVe, 2015.
[4] Chunjiao Dong, Stephen H. Richards, Baoshan Huang, and Ximiao Jiang. Identifying the factors contributing to the severity of truck-involved crashes. *International Journal of Injury Control and Safety Promotion*, 22(2):116–126, 2015;2013;.
[5] PwC. Look mom, no hands!, 2013.
[6] C. Urmson and W. ". Whittaker. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2):66–68, March 2008.
[7] Houston Chronicle. Get ready for automated cars, 2012.
[8] Dudley David. The driverless car is (almost) here, 2015.
[9] Programme for a european traffic system with highest efficiency and unprecedented safety, 1987-1995.
[10] Robert D Leighty. Darpa alv (autonomous land vehicle) summary. 1986.
[11] Pomerleau A. Dean. Alvinn: an autonomous land vehicle in a neural network. 1989.
[12] DARPA. Darpa grand challenge, 2004;2005;2007.
[13] Braunschweig University of Technology. Carolo cup competition, 2016.
[14] U.S. Government. Gps accuracy.
[15] Heba Aly, Anas Basalamah, and Moustafa Youssef. Robust and ubiquitous smartphone-based lane detection. *Pervasive and Mobile Computing*, 26:35 – 56, 2016.
[16] Barbara Kitchenham, Rialette Pretorius, David Budgen, O. Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. Systematic literature reviews in software engineering – a tertiary study. *Information and Software Technology*, 52(8):792–805, 2010.
[17] Christian Berger. Opendavinci - open source development architecture for virtual, networked, and cyber-physical system infrastructures.
[18] Nico Boh Christian Berger. Carolo cup: Rules and regulations, 2014.
[19] Andy Dingley. Ackermann steering geometry, 2010.
[20] Sibel Yenikaya, Gökhan Yenikaya, and Ekrem Düven. Keeping the vehicle on the road: A survey on on-road lane detection systems. *ACM Computing Surveys (CSUR)*, 46(1):1–43, 2013;2014;.
[21] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, 2014.
[22] Gurveen Kaur and Dinesh Kumar. Lane detection techniques: A review. *International Journal of Computer Applications*, 112(10), 2015.
[23] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.
[24] Richard Duda and Peter Hart. Use of the hough transformation to detect lines and curves in pictures, 1972.
[25] Martin Fischler and Robert Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, 1981.
[26] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.
[27] Yajun Yang. Polynomial curve fitting and lagrange interpolation. *Mathematics and Computer Education*, 47(3):224, 2013.
[28] Yue Wang, Eam K. Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision Computing*, 22(4):269–280, 2004.
[29] Cláudio R. Jung and Christian R. Kelber. Lane following and lane departure using a linear-parabolic model. *Image and Vision Computing*, 23(13):1192–1202, 2005.
[30] M. Aly. Real time detection of lane markers in urban streets. pages 7–12. IEEE, 2008;2014;.
[31] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26, 2008.
[32] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, and Huiyan Chen. A novel lane detection based on geometrical model and gabor filter. pages 59–64, 2010.
[33] Qing Lin, Youngjoon Han, and Hernsoo Hahn. Real-time lane departure detection based on extended edge-linking algorithm. pages 725–730, 2010.

[34] F. Mariut, C. Fosalau, and D. Petrisor. Lane mark detection using hough transform. pages 871–875. IEEE, 2012.
[35] K. Ghazali, Rui Xiao, and Jie Ma. Road lane detection using h-maxima and improved hough transform. pages 205–208. IEEE, 2012.
[36] Yingmao Li, Asif Iqbal, and Nicholas R. Gans. Multiple lane boundary detection using a combination of low-level image features. pages 1682–1687. IEEE, 2014.
[37] Sukriti Srivastava, Ritika Singal, and Manisha Lumba. Efficient lane detection algorithm using different filtering techniques. *International Journal of Computer Applications*, 88(3), 2014.
[38] Amol Borkar, M. Hayes, M. T. Smith, KTH, Skolan för informations- och kommunikationsteknik (ICT), and CoS Kommunikationssystem. A novel lane detection system with efficient ground truth generation. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):365–374, 2012.
[39] Jae-Hyun Cho, Young-Min Jang, and Sang-Bock Cho. Lane recognition algorithm using the hough transform with applied accumulator cells in multi-channel roi. pages 1–3. IEEE, 2014.
[40] Chan Y. Low, Hairi Zamzuri, and Saiful A. Mazlan. Simple robust road lane detection algorithm. pages 1–4. IEEE, 2014.
[41] Jun Wang, Tao Mei, Bin Kong, and Hu Wei. An approach of lane detection based on inverse perspective mapping. pages 35–38. IEEE, 2014.
[42] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, 30(1):32–46, 1985.
[43] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.
[44] Alan R Weiss. Dhrystone benchmark: History, analysis, scores and recommendations. 2002.
[45] OpenCV Docs. How to scan images, lookup tables and time measurement with opencv.
[46] Per Runeson, Martin Höst, Institutionen för datavetenskap, LTH Faculty of Engineering, Lunds universitet, Department of Computer Sciences, Institutioner vid LTH, Lunds Tekniska Högskola, Lund University, and Departments at LTH. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.