To appear in: Proceedings of 12th IEEE/IET International Symposium on Communication Systems, Networks, and Digital Signal Processing (CSNDSP), 20-22 July 2020, Porto, Portugal

System Design for a Driverless Autonomous Racing Vehicle

J. Culley, S. Garlick, E. Gil Esteller, P. Georviev, I. Fursa, I. Vander Sluis, P. Ball and A. Bradley*

School of Engineering, Computing & Mathematics, Oxford Brookes University

The rising popularity of autonomous vehicles has led to the development of driverless racing cars, where the competitive nature of motorsport has the potential to drive innovations in autonomous vehicle technology. The challenge of racing requires the sensors, object detection and vehicle control systems to work together at the highest possible speed and computational efficiency. This paper describes an autonomous driving system for a self-driving racing vehicle application using a modest sensor suite coupled with accessible processing hardware, with an object detection system capable of a frame rate of 25fps, and a mean average precision of 92%. A modelling tool is developed in open-source software for real-time dynamic simulation of the autonomous vehicle and associated sensors, which is fully interchangeable with the real vehicle. The simulator provides performance metrics, which enables accelerated and enhanced quantitative analysis, tuning and optimisation of the autonomous control system algorithms. A design study demonstrates the ability of the simulation to assist in control system parameter tuning - resulting in a 12% reduction in lap time, and an average velocity of 25 km/h - indicating the value of using simulation for the optimisation of multiple parameters in the autonomous control system.

Keywords: Autonomous vehicle, driverless car, vehicle control, vehicle simulation, autonomous vehicle simulation.

1. Introduction

Approximately 94% of all road accidents are caused by human error [1]. Autonomous Vehicles (AVs) can 'see' in multiple directions at once, and are robust against distraction - thus they have the potential to save many lives. Much of the development of Autonomous Vehicle (AV) technology necessitates testing within a sandboxed environment away from pedestrians and other road users, and thus the closed environment of a racetrack provides an ideal location for testing new technologies and algorithms. In addition, the fierce competition within the Motorsport industry has a long history of rapidly enhancing technology development [2], and the high speeds and narrow margins involved in racing push the limits of the vehicle, sensors and control system, and thus drive advances in self-driving technology which can benefit road safety.

There is a wide body of literature detailing a variety of approaches to autonomous vehicle control [3, 4, 5, 6, 12]. Mihalea et al. [3] aim to provide driver assistance using an 'end-to-end' approach - where the entire process of driving is a learned transformation from image to output. Whilst there are many potential benefits to this approach, the application in a complete vehicle control system requires a large amount of training data, and generates a model where the reasons behind a certain vehicle response are essentially unknown.

A more common approach to autonomous robotics is that the problem is broken down into a pipeline of subsystems including; Perception, Decision, Control and Actuation [4]. Perception in particular provides several areas where research is required; accurate object detection (e.g. road signs, traffic cones, other vehicles, pedestrians etc), localisation (identifying where the vehicle is) and mapping (generating a map of the environment). Localisation and mapping are commonly carried out simultaneously, enabling the AV to generate or update a map as it progresses through an environment.

Tian et al. [5] provide an overview of the complete vehicle, system and processing pipeline required for an autonomous racing vehicle with experimental results, while Andresen et al. [6] give a detailed methodology for the development of a fast, accurate mapping subsystem with a detailed performance assessment.

While the studies outlined above focus on experimental testing, there is an emerging trend of the development of AV simulators. Ahamed et al. [7] propose the use of simulation to evaluate and compare AV control algorithms - thus bridging

* Corresponding author. Email: abradley@brookes.ac.uk

the gap between design and real-world testing, and thereby accelerating the development process. Specifically, the study develops a model using Gazebo [8] software. Shah et al. [9] detail the vehicle model, physics behaviour and modelling of sensors including gyroscope, accelerometer and GPS, with noise models that enable the simulator to mimic the behaviour of a real vehicle - though in this case, the vehicle is a drone.

The use of simulation also facilitates the knowledge of a 'ground truth' - an absolute reference of the underlying reality - with complete certainty, which can be difficult (or expensive) to achieve with real-world testing. This enables the vehicle's perception of the world to be compared against the ground truth, thereby enabling the performance of the AV control system to be quantified and compared in a variety of different scenarios [10]. In addition, the ability to repeatedly simulate the same event allows various parameters of the control system to be optimised [11].

There has been little reported work providing an integrated approach to the development of a real-world AV control system and the ability to interchange the real vehicle for a simulation - thus enabling quantitative analysis, assessment and optimisation of the control system. The development of this holistic approach requires the synthesis of hardware, control algorithm, simulation and sensor modelling research - providing a significant challenge for those wishing to embark upon autonomous vehicle development.

Kabzan et al. [12] address this knowledge gap by providing a detailed overview of a fully autonomous control system for the specific application of the Formula Student Driverless competition - including the application of the system in a Gazebo-based simulator. The system utilises a range of sensors including LiDAR, 3 cameras, a laser ground-speed sensor and a custom-designed processing unit. Whilst the AV control system is a comprehensive solution, the simulation does not include models of the sensors - and thus the complete AV pipeline is not being fully tested. In addition, the use of simulation for control system parameter tuning and optimisation is not discussed.

This paper provides an overview of an autonomous driving system for a self-driving racing vehicle application using a more modest sensor suite coupled with accessible processing hardware, and a modelling tool developed in open-source software for real-time dynamic simulation of the autonomous vehicle and associated sensors. The simulator enables accelerated and enhanced quantitative analysis, tuning and optimisation of the self-driving algorithms in operation within the vehicle. As the simulator mimics the vehicle and sensor behaviour, it is fully interchangeable with the real car. This facilitates AV development using only a PC - without the need for access to a real vehicle or any sensor hardware.

This holistic approach provides a methodology which helps to lower the barrier for entering the rapidly-advancing field of autonomous vehicle development.

2. Methodology

A full self-driving system is realised by using an array of sensors including a stereo camera, GPS, Inertial Measurement Unit (IMU) and Wheel Speed Sensors (WSS).

An Artificial Neural Network object recognition algorithm is used to identify the objects defining the track boundaries, which is combined with depth perception to ascertain the position of the track boundaries from the vehicle's perspective. A path planning algorithm then identifies a target line to follow between the lines.

To enhance the performance of the vehicle a mapping algorithm has been developed to build, record and store historic data during the first lap of the track - enabling previously-seen track boundaries which are currently outside the current field of view to be known, and subsequent laps to be undertaken with a knowledge of the upcoming trajectory. A localisation and state estimation algorithm is employed to fuse the GPS, IMU and wheel speed data, thus enabling the vehicle to translate the ego-centric perception of track boundary information into a global map of the racetrack.

The vehicle can operate in one of two modes; ego-centric - where the objects demarking the track are instantaneously joined to form a route ahead of the vehicle; and global - where the objects are identified and joined on a global map. This facilitates an initial low-speed approach such that the vehicle can drive an unknown circuit, and a high-speed approach once it has prior knowledge of the circuit. The ego-centric method does not require localisation - and thus is robust to failures of navigation equipment.

From these track boundaries a target line is identified, and a driver model uses this to generate vehicle control requests which are delivered to the car via CAN bus.

Finally, an autonomous vehicle simulation has been developed, including a vehicle dynamic model, sensor models, simulated camera visuals and the control system itself, which is used to demonstrate the ability of the autonomous control system to complete a lap of the circuit. The simulation provides identical inputs and outputs to the real sensors and CAN bus network - thus the simulator is fully interchangeable with the real car.

The simulation allows quantification of the control system performance against the ground truth, and hence this tool offers the potential for quantitative analysis of AV control system performance, parameter tuning and optimisation, providing a baseline for real-world experimental testing.

A. System overview

The topology of the autonomous control system which includes sensors, processing and control is shown in Fig 1.

The system comprises a ZED stereo camera, Peak PCAN IMU, GPS and wheel speed sensors. Processing is undertaken using an InCarPC CQ67G vehicle-mounted PC equipped with an Intel Core i7-6700TE with 16GB RAM and an NVIDIA GTX1050 TI GPU.

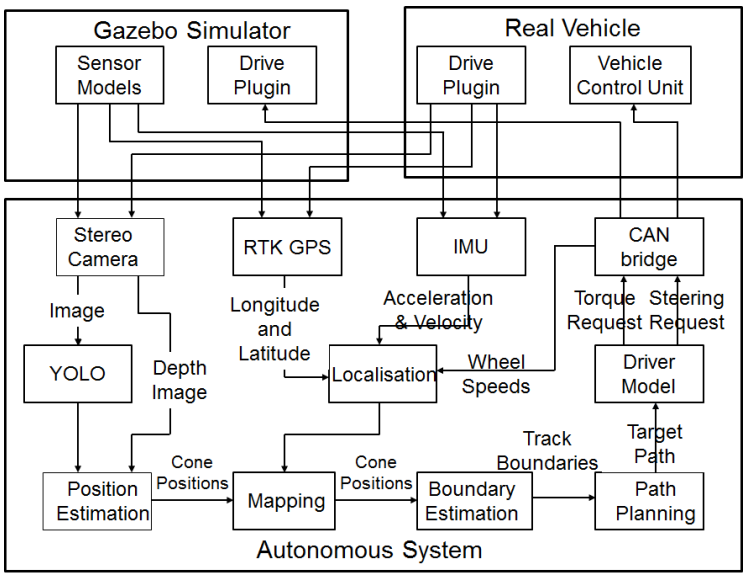

Fig. 1 System overview

The system uses ROS 2 - Dashing for inter-process communication. This provides a distributed approach, which is more robust, enables future expansion of the system and ease of development. ROS 2 offers support for real time control and improvements on inter-machine communication.

B. Perception

Whilst driving around the track in any given event, the car needs to perceive and understand the surrounding environment, to enable the vehicle to drive the track at racing speeds. The circuit is marked out with coloured traffic cones, and therefore the vehicle must, in real time, detect the colour and 3D position of each cone to build a representative map, enabling the car to understand and race the current track. This system uses the stereo camera for colour imaging and depth estimation. This enables the car to use object detection, utilising YOLOv3 Tiny [13] object recognition on colour images of a reduced resolution, to find key objects of interest such as the different coloured cones delineating the track boundaries. Once the location of each object is identified, the depth map from the stereo camera is used to build an ego-centric map of the locations of the cones with respect to the vehicle. Image processing is initially carried out at a resolution of 608x352 pixels, to achieve a balance between object detection quality and frame rate. The dataset used for training the algorithm contains over 1000 manually labelled images, containing at least 10000 annotations captured with the camera as used on the vehicle. A number of augmentations were applied to the dataset during training in order to diversify the images.

C. Localisation and mapping

All sensors inherently introduce some degree of error [14], and thus any simultaneous localisation and mapping (SLAM) solution must be robust under conditions of uncertainty. Localisation is therefore executed using data provided by a variety of sensors; the GPS, IMU, and WSS. The wheel speed sensors estimate the longitudinal velocity of the vehicle, while the

IMU measures the acceleration and rotation of the vehicle in all directions. The GPS provides an approximate location for the vehicle up to a few metres. All of this information is combined into an Extended Kalman Filter (EKF) [15], which calculates the state of the vehicle by determining its most probable location.

To address conditions of uncertainty in regards to mapping, the locations of the cones around the car are repeatedly sampled. Machine Learning is employed to identify each unique object from the samples by K-means clustering [16], treating each sampled location of a given cone as a point in a cluster, the centre of which is estimated as the location for that cone. This position is added to the global map of the circuit, and continues to be incrementally refined as more observations become available.

In order to be considered reliable, the cluster must reach a minimum number of observations. As the number of observations increases, the value of each additional point in a cluster diminishes - and thus system performance can be improved by 'locking' clusters to a fixed size once they pass a threshold. Locking clusters is computationally beneficial as once locked, they no longer require recalculation.

The disadvantage of this approach is that it relies heavily upon the accuracy of the GPS, IMU & WSS-based navigation - if there is an error in this calculation (e.g. caused by a loss of GPS accuracy), the map will reduce in accuracy. In order to address this, it is possible to enhance the accuracy of the localisation by using the map itself to estimate the position of the vehicle using a particle filter.

The particle filter works by generating a number of random positions (or particles) and predicting the objects that would be perceived if the vehicle was at that position - the particle with the closest approximation to the actual detected cones is then used, and the process repeated with a higher density of particles in the best locations [16].

Once a map (or partial map) of the layout of objects delineating the track is known, this information is passed to the boundary estimation algorithm to generate a track map.

D. Track boundary estimation

In order to plan a path for the car it is important to first understand the track. Estimation of the track boundaries can be carried out in either ego-centric or global mode - though they operate in an identical manner - essentially joining each cone to the next along the circuit.

Through testing, algorithms such as nearest neighbour [17] were found to be insufficient for this task. Where cones were spaced farther apart than the track is wide, it would often result in track boundaries being formed in front of the car perpendicular to the track. In most cases it was observed that the actual track boundary was equivalent to the shortest path between the points, more commonly known as the traveling salesman problem [18], and that the track boundary was the 'straightest' line connecting all cones (of a particular colour) together. The estimation of this line was achieved using an adaptation of the nearest neighbour algorithm which introduced an additional heuristic.

The algorithm begins by selecting any two cones and iterating through cones within the vicinity which are ranked using a heuristic based upon the angle and the distance between these sets of three points. By continuously joining up the points in this manner a track boundary estimation is created which is suitable for most tracks, thus enabling the path planning to calculate a path. However, this path planning model only works for tracks where there is a single route, meaning this algorithm fails for tracks with dividing or overlapping paths, such as a pitlane junction or figure of eight. To address this, a heuristic based upon the distance of a point and the ratio of its length compared to all other points was created. It was found, through testing various scenarios, that selecting the closest points with the most similar length ratio provides a suitable boundary estimation.

E. Path planning

The basic function of the path planning algorithm is to calculate the midline between the track boundaries, enabling the vehicle to complete a lap of the circuit. A minimum curvature path can also be used – i.e. a basic racing line.

The algorithm implemented in this system is able to predict a trajectory by calculating a set of points at fixed 0.5m intervals ahead of the car which form the trajectory for the car to follow. The points can be biased to direct the points to tend towards a specified direction, in turn biasing the direction of the car. This ability to force the algorithm to produce a path that can be directed is crucial to solving the figure-of-eight problem, in which the car must choose which route to select. It is this biasing of the line that allows the car to choose which path it takes.

The set of points for the trajectory can be formed by producing a series of lines that intersect the track boundaries on each side. Theoretically, the line with the closest intersection points is the most perpendicular line to the track and is therefore a waypoint which the car should pass through. By iterating on these steps we can form a set of points which forms the trajectory for the car. By biasing where on each waypoint the target point lies, we can direct the car to either follow the left, right or centre of the track. This simple solution to trajectory selection allows the car to plot a path through an intersection.

F. Vehicle control

Vehicle control is obtained using a driver model. The model selected is an explicit path-following driver. In this approach, real-time calculation is used to determine the tracking error of the vehicle with respect to the path.

Processing signals are obtained from various sensors and the odometry, relating the measurements with the ground truth of the path. The driver model for this system follows the points generated by the path planning sub-system. The controller comprises two parts; longitudinal, which controls the velocity of the car, and lateral, which simulates the response of the driver's input to the steering system.

A Proportional-Integral-Derivative (PID) controller is used for longitudinal control; regulating throttle based upon a target velocity - which depends inversely upon the upcoming corner radius. The lateral controller also uses a PID controller, which attempts to minimise the error over by adjusting the steering input. It is based upon a Pure Pursuit Controller [19], computing angular velocity and moving the vehicle from the actual position to reach a look-ahead point. The look-ahead distance is adjusted as a percentage of the available target path from the perception system.

For evaluation of the driver model performance, any deviation from the target path is seen as an error, and a larger deviation that takes the driver outside an allowed margin for error is considered a collision [20] - or in this case an excursion from the track boundary.

G. Simulating an autonomous vehicle

The aim of the simulation is to create a digital twin of the vehicle, sensors and track environment to facilitate testing, analysis, tuning and optimisation of the complete autonomous control system. Gazebo has been used for this purpose - this is an open-source 3D robotics simulator integrating ODE physics engine, OpenGL rendering, and support code for sensor simulation and actuator control [8].

Virtual world generation was achieved through creation of 3D CAD models of items including track elements (e.g. cones, barriers, ground lines etc) and other environmental models including trees, pedestrians or even other vehicles.

Combining Gazebo with ROS 2 and Python, the vehicle dynamic behaviour has been modelled. It is based upon a bicycle model [76] with simplified input parameters. The physical engine included in Gazebo is used to compute weight transfer and inertia effects. Camber and suspension effects are neglected. A soft racing tyre has been modelled, using lateral and longitudinal force coefficients of 1.5.

The simulated vehicle includes a stereo camera, depth camera, LiDAR, GPS, IMU and wheel speed sensors (though the LiDAR data is not utilised in the AV control system presented in this paper). The camera images offer a high resolution, undistorted image. Simple sensor models are applied to each in a similar manner to Shah et al. [9], such that the sensors exhibit noise and other characteristics - thus mimicking the behaviour of real sensors. In addition to the sensor data, the simulation outputs the ground truth data - facilitating a comparison between the simulated world, and that perceived by the AV control system.

The simulator is fully interchangeable with the real vehicle, and thus it is possible to test and develop every aspect of the autonomous control system in a virtual environment. In addition, the quantification of performance offered by comparison to a ground truth facilitates tuning and optimisation of the control system parameters.

3. Results and discussion

The perception system is capable of processing images at a frame rate of 25fps, and accurately detecting cones. Measurements indicate that 87% of detections are accurate, and that 93% of all cones in a frame are detected (recall). This delivers a mean average precision of 92%. These metrics compare favourably with Dhall *et al.* [21] who also reported a lower frame rate (although this is likely due in part to the difference in hardware implementation).

In the current implementation, the precision and recall have been sacrificed in favour of increasing the processing speed by downsizing the resolution. This results in more distant objects (~15-20m away) being reduced to a small number of pixels, and thus they become difficult or impossible to detect, thereby reducing both precision and recall. At short range, the performance metrics are considerably higher than those quoted above. Identified cones at a variety of distances from the vehicle are depicted in Fig 2.

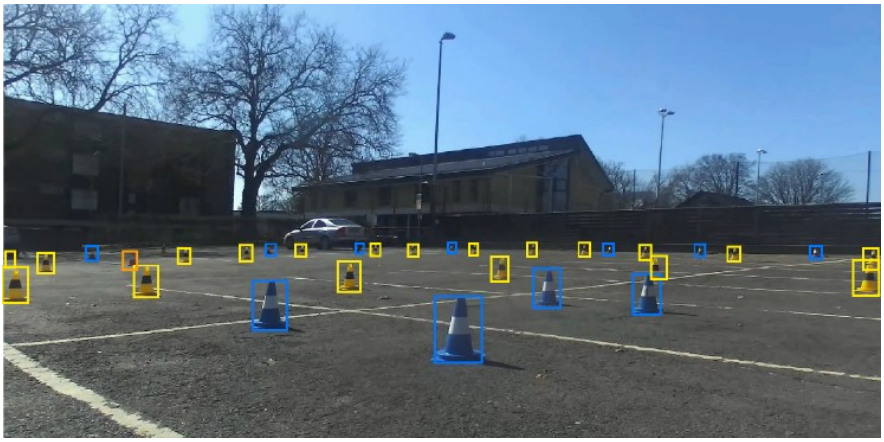

Fig 2. Inference during real-world testing using trained CNN

The simulation of the autonomous vehicle is depicted in Fig 3., with the complete AV control system running on data provided by the sensor models. The perception system (top left) is detecting cones, with boundaries being detected, and an ego-centric trajectory being planned (top right).

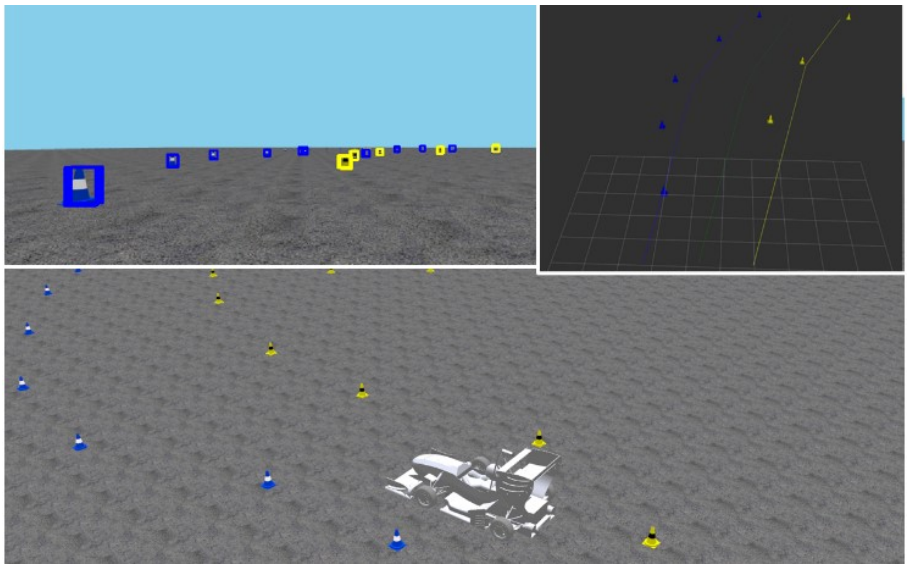

Fig 3. Gazebo simulation of autonomous vehicle (lower), perception system identifying cones (inset left) and estimated track boundaries (inset right)

The use of this simulation enables a quantitative analysis of the ability of the entire AV control system to detect, identify and follow a target path around the circuit. The error between the vehicle's position and the ideal line (in this case the midline, though this could later be obtained from an optimal path) can then be evaluated (Fig. 4). This information enables a detection of any points at which the vehicle leaves the track boundary (e.g. at ~25m from the start line), and a calculation of the RMS error.

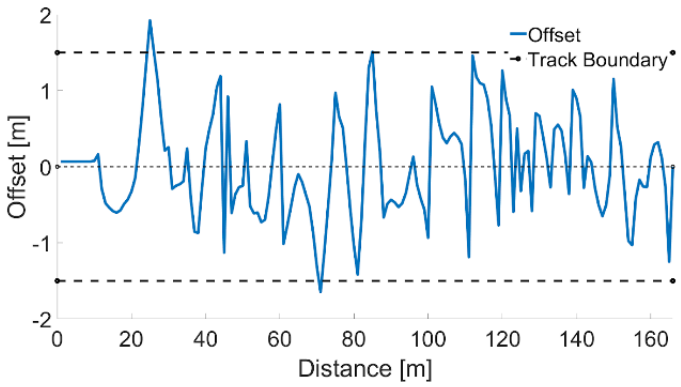

Fig 4. Vehicle Offset with respect to the target path

Table I shows an example of the simulation output metrics of a single lap of the circuit (with non-optimal parameters), giving an RMS path error of 0.35m.

Table 1. Example simulation output metrics

Average velocity [km/h]	RMS error [m]	Lap time [s]
19.2	0.35	53.01

The effect of adjustments to each parameter (in this example look-ahead distance and PID controller gain) upon the actual path driven can be plotted on a map of the circuit as shown in Fig 5 - thus enabling visualisation of areas where the vehicle is struggling to follow the target path. The segments of the simulation where issues occur can then be replayed to ascertain exactly what the behaviour of each aspect of the control system was at these instances - which is invaluable for debugging the algorithms.

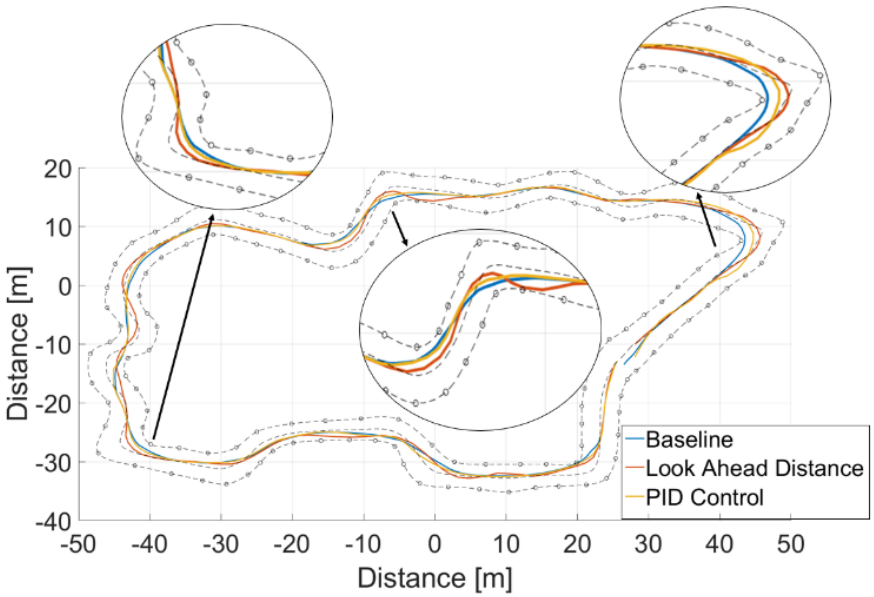

Fig 5. Vehicle path taken after adjusting the controller

A benefit of the use of simulation is the ability to run repeated simulations with an automated adjustment of the vehicle parameters. Fig 6. shows an example of the results of a simple design study delivering a 6 second per lap (or 12%) reduction in lap time when investigating the effect of adjusting a single AV control parameter (in this case the driver model look-ahead parameter) on lap time. This equates to an average velocity of 25 km/h.

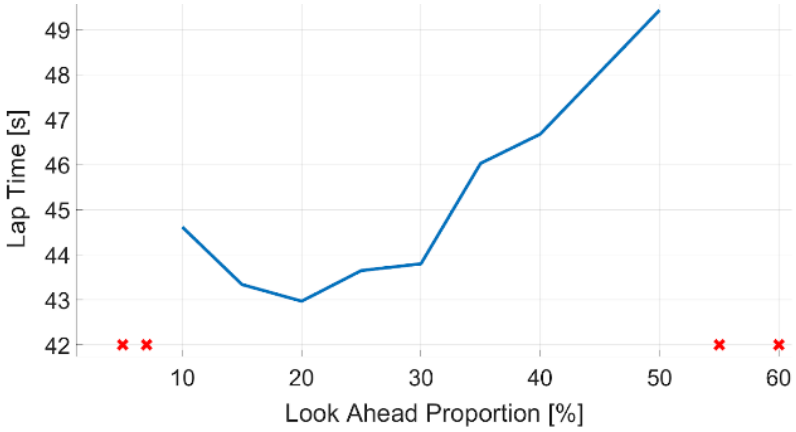

Fig 6. Effect of control system parameter variation upon lap time:
X denotes leaving the track boundaries, and thus failing to complete a lap

This automated parameter adjustment, simulation and analysis technique enables thousands of lap simulations to be conducted in an unsupervised manner - a process common in traditional, human-driven motorsport for vehicle setup optimisation [22]. This enables the parameter values to be identified that result in the minimum possible lap time whilst remaining with the track boundaries.

Future work will focus on the tuning of parameters including; camera mounting locations, perception system resolution, path planning parameters and driver model PID control system gain values with the aim of reducing trajectory error and increasing average velocity - thus forming a baseline setup for experimental testing.

4. Conclusions

An autonomous driving system for a self-driving racing vehicle application has been described using a modest sensor suite and accessible processing hardware. The system is capable of identifying objects delineating a track boundary, mapping the environment, planning a route and delivering control inputs to a vehicle over CAN bus. In experimental testing the perception system demonstrated the ability to process images at a frame rate of 25fps, with 87% precision, recall of 93%, and a mAP of 92% - which compares favourably with other similar work.

A simulation modelling tool, developed in open-source software, for real-time dynamic simulation of the vehicle and associated sensors is described. The simulation is fully interchangeable with the real vehicle, thus facilitating algorithm development using only a PC - without the need for access to a real vehicle or any sensor hardware. The use of simulation enables accelerated and enhanced quantitative analysis, tuning and optimisation of the self-driving algorithms in operation within the vehicle. Overall vehicle performance metrics including average velocity, path error, excursions from track boundaries and lap time are quantified. Initial results achieved an RMS path error of 0.35m. A design study was conducted to demonstrate the ability of the simulation to assist in tuning the AV control system parameters - resulting in a 12% reduction in lap time - and indicating the potential of using simulation for further parameter optimisation.

The self-driving system is capable of completing a lap of the simulated test track at an average velocity of 25 km/h, however further work is required to evaluate the interdependencies of the key parameters and determine the optimum - which will later be used as a baseline for real world testing and experimentation.

The holistic approach presented in this paper provides a methodology which helps to lower the barrier for entering the rapidly-advancing field of autonomous vehicle development.

Acknowledgment

This project has been made possible with the support of the School of Engineering, Computing & Mathematics at Oxford Brookes University. The authors would like to thank; Matt Woolford for his work on perception; Christian Strang for the localisation; and Fabio Cuzzolin, Matthias Rolf, Alex Rast and Gordana Collier for their support and direction. Credit goes to the following sponsors; OXTS and Datron for their support and expertise in vehicle localisation; Parkopedia for their invaluable advice, guidance and assistance; and StreetDrone for providing continued support throughout the project.

References

[1] US Department of Transportation, National Highway Traffic Safety Administration, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey", Traffic Safety Facts, No. DOT HS 812 115, Feb 2015

[2] J-P Skeete, "The obscure link between motorsport and energy efficient, low-carbon innovation: Evidence from the UK and European Union", .Journal of Cleaner Production, Volume 214, pp. 674-684, 2019

[3] A Mihalea ; R Samoilescu ; A Cristian Nica ; M Trăscău ; A Sorici ; A Magda Florea, "End-to-end models for self-driving cars on UPB campus roads", IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, Romania, 5-7 Sept. 2019

[4] J.P. Giacalone, L. Bourgeois, and A. Ancora, "Challenges in aggregation of heterogeneous sensors for Autonomous Driving Systems", IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 1-13 March 2019

[5] Tian et al., "Autonomous Driving System Design for Formula Student Driverless Racecar", IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26-30 June 2018

[6] L.Andresen et al., "Fast and Accurate Mapping for Autonomous Racing", arXiv:2003.05266 [cs.RO], March 2020

[7] Ahamed et al. "Software-in-the-loop Modeling and Simulation Framework for Autonomous Vehicles", IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3-5 May 2018

[8] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 Sept.-2 Oct. 2004, pp 2149-2154

[9] S Shah, D Dey, C Lovett, A Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: Hutter M., Siegwart R. (eds)", Field and Service Robotics, Springer Proceedings in Advanced Robotics, vol 5. Springer, Cham, 2018

[10] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator", arXiv:1711.03938 [cs.LG], Nov 2017

[11] Dai et al. "Control Parameter Optimization for Autonomous Vehicle Software Using Virtual Prototyping", IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Toulouse, France, 23-26 Oct. 2017

[12] J. Kabzan et al "AMZ Driverless: The Full Autonomous Racing System", arXiv.org, arXiv:1905.05150, May 2019

[13] J. Redmon, A. Fahradi "YOLOv3: An Incremental Improvement", arXiv:11804.02767 [cs.CV], April 2018

[14] V. Fox, J. Hightower, Lin Liao, D. Schulz, G. Borriello. (July 2003). "Bayesian filtering for location estimation". In: IEEE Pervasive Computing 2.3, pp. 24-33. DOI: 10.1109/MPRV.2003.1228524.

[15] S.Thrun, W. Burgard and D.Fox, "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)". The MIT Press, 2005

[16] J.B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1. University of California Press, 1967, pp. 281–297

[17] Leif E. Peterson (2009) K-nearest neighbor. Scholarpedia, 4(2):1883.

[18] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", Wiley, ISBN: 978-0-471-90413-7 August 1985

[19] R. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm". The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Jan 1992

[20] A. Levesque and J.Johrendt, "The State of the Art of Driver Model Development" SAE Technical Paper 2011-01-0432, 2011

[21] A. Dhall, D. Dai, L. V. Gool "Real-time 3D Traffic Cone Detection for Autonomous Driving", IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9-12 June 2019

[22] I. Colunga, A. Bradley "Modelling of transient cornering and suspension dynamics, and investigation into the control strategies for an ideal driver in a lap time simulator", Proc. IMechE, Part D: Journal of Automobile Engineering, 228(10) pp.1185-1199, 2014