RESEARCH PROJECT IN MECHATRONICS ENGINEERING

REINFORCEMENT LEARNING FOR LOCAL PATH FOLLOWING OF AN AUTONOMOUS F:SAE VEHICLE

Harvey Merton

Project Report ME16-2021

Co-worker: Thomas Delamore

Supervisor: Dr Karl Stol

Department of Mechanical Engineering
The University of Auckland

15 October 2021

ME16-2021

REINFORCEMENT LEARNING FOR LOCAL PATH FOLLOWING OF AN AUTONOMOUS F:SAE VEHICLE

Harvey Merton

ABSTRACT

Formula:Society of Automotive Engineers (F:SAE) is a series of international competitions where students design, manufacture and race their own formula-style race cars. With the impending introduction of a driverless class to the Australasian competition, University of Auckland's F:SAE team need to begin research into autonomous capability.

This project aims to use deep reinforcement learning to map locally-observed cone positions to a desired steering angle on a vehicle travelling at constant speed. The soft actor critic (SAC) and adversarial inverse reinforcement learning (AIRL) algorithms are used for this purpose, which no other F:SAE team has yet published literature on. These algorithms are first used to train models in a custom ROS-based simulation (converging in 735 and 1386 episodes respectively). The models are then tested in inference for track completion repeatability.

Inference testing occurs first in simulation and then in a physical test setup. Simulation results show that both SAC and AIRL-trained models achieve a 100% median completion rate in simulation on the training track. On a different track, AIRL models achieve 100% median completion while SAC only sees a 70% median completion rate. In physical tests, AIRL achieves a full track completion rate of 92.5%, while SAC achieves 87.5%.

Comparing the mean steering angle derivative demanded from SAC of $57.6°/s^2$ and AIRL of $56.9°/s^2$ to a hard-coded pure pursuit expert ($4.4°/s^2$), neither algorithm is smooth enough for direct transfer to a real F:SAE vehicle yet. However, both algorithms have proven feasible without the need of complex dynamics modelling and with further work, may perform comparably to, or even outperform, traditional methods.

DECLARATION

Student

I . Harvey Merton ..... hereby declare that:

1. This report is the result of the final year project work carried out by my project partner (see cover page) and I under the guidance of our supervisor (see cover page) in the 2021 academic year at the Department of Mechanical Engineering, Faculty of Engineering, University of Auckland.
2. This report is not the outcome of work done previously.
3. This report is not the outcome of work done in collaboration, except that with a project sponsor as stated in the text.
4. This report is not the same as any report, thesis, conference article or journal paper, or any other publication or unpublished work in any format.

In the case of a continuing project: State clearly what has been developed during the project and what was available from previous year(s):

N/A

Signature: 

Date: 13/10/21

Supervisor

I confirm that the project work undertaken by this student in the 2021 academic year ~~**is /**~~ **is not** (*strikethrough as appropriate*) part of a continuing project, components of which have been completed previously.

Comments, if any:

Signature: 

Date: 14/10/21

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Karl Stol for his continued support throughout the project. His expertise, eye for detail, swift responses and logical thinking were critical in this project's success. Further, through his feedback, my research skills have developed exponentially.

It was a pleasure to have Thomas Delamore as my partner. His positivity, tenacity, collaborative spirit and good humour made the project all the more enjoyable.

I extend huge appreciation to Henry Williams who, despite not being our official supervisor, was always willing to help out. His expertise in robotics, ROS and stereo cameras was much needed and appreciated.

Further thanks goes to both Logan Stuart and Emanuele Romano, the mechatronics lab technicians. Their help ordering components, maintaining 3D printers and providing tools and materials made the technical implementation process much smoother.

Lastly I would like to thank Zara Delamore for being our test kart driver. Without her help, it would have been impossible to gather our physical results.

Glossary of Terms

Action	What a RL model selects to perform on an environment.
Agent	An entity that performs actions on an environment.
Algorithm	Refers to RL algorithms that train RL models.
(Stereo) camera	Two cameras separated by a small fixed distance acting together to localise objects.
Domain adaptation	A method where an RL model is trained in simulation and trained further in the physical world.
Domain randomisation	Changing simulation parameters during training to produce a more robust RL model.
Environment	Specific parts of a (simulated or physical) world that an RL model interacts with. Refers to the FS simulator in this text.
Episode	The set of time-steps from when the environment resets until the model completes or fails a task. Multiple episodes occur when training.
Expert	An agent that performs almost perfect actions in an environment.
F:SAE-47	The University of Auckland's F:SAE team.
Inference	The second stage in RL where a trained model is used to select actions.
Kart	A pedal go-kart used for physical world testing.
(ArUco) marker	A specific black and white pattern with a unique identification number that can be easily localised using computer vision.
Model	The output of a RL algorithm - contains the policy.
Model-free	No pre-defined mathematical description of an entity or environment is required.
Off-policy	Random actions rather than the current policy is used to explore the environment during training.
Policy	A method for determining the optimal action.
Pose	Position and orientation.
Time-step	The length of time that a selected action is performed on an environment before a new action is selected.
Track	A path delineated by coloured cones.
Training	The first stage of RL where a policy is learned to maximise rewards from an environment. A model is the output.
Transfer learning	Training a model further on a new environment that has already been trained on another environment.
Reinforcement Learning	A machine learning method where an agent trains itself through performing a series of trials on an environment.
(Steering angle) rate limit	The maximum speed at which the steering angle can be turned.
Reward	A number returned from an environment to indicate the effectiveness of an action selected during training.
Reward function	A function defined to generate the reward.
Steering angle	The angle of the steering wheel from straight.

Abbreviations

AIRL	Adversarial Inverse Reinforcement Learning
AMZ	Academic Motorsports Club Zurich
ANN	Artificial Neural Network
DRL	Deep Reinforcement Learning
FS	Formula Student
FSG	Formula Student Germany
F:SAE	Formula Society of Automotive Engineers
F:SAE-A	F:SAE Australasia
IRL	Inverse Reinforcement Learning
IQR	Inter-Quartile Range
ML	Machine Learning
RL	Reinforcement Learning
ROS	Robot Operating System
SAC	Soft Actor Critic
SAE	Society of Automotive Engineers

Nomenclature

{X, Y, Z} Local Cartesian co-ordinate system in simulation with origin fixed to the vehicle chassis' centre

{x, y, z} Global Cartesian co-ordinate system in simulation with origin at the vehicle reset position

{r, θ} Local polar co-ordinate system in simulation with origin fixed to the vehicle chassis' centre

$\mu_{r,\theta}$ Mean cone position in {r, θ} co-ordinates

$\sigma_{r,\theta}$ Standard deviation of cone position in {r, θ} co-ordinates

α_{1-7} Reward function scaling parameters

1. Introduction

Formula:Society of Automotive Engineers (F:SAE), also known as Formula Student (FS) in some regions, is a series of international events that challenge university teams to design, manufacture and race a fully-custom, formula-style race car (see Figure 1). The University of Auckland Formula:SAE team (F:SAE-47) is one F:SAE team who compete annually in the Formula:Society of Automotive Engineers-Australasia (F:SAE-A) competition held in Australia in December. This competition is currently split into two classes that compete independently: combustion vehicles and electrical vehicles.

Figure 1 F:SAE-47's 2019 electric vehicle.

In 2017, Formula Student Germany (FSG) became the first F:SAE competition to introduce a driverless vehicle class - called Formula Student Driverless (FSD) (see [1] for 2022 rules and Appendix A for a points and events breakdown). Vehicles in this class must autonomously navigate tracks (see Appendix B for examples) delineated with the cones seen in Figure 2. At the 2019 F:SAE-A event, two Australian teams (Monash Motorsport and University of Technology Sydney) showcased some autonomous capability with the intention of creating interest in a driverless event at F:SAE-A. To continue to be competitive, F:SAE-47 should begin the development of an autonomous vehicle now in preparation for the inevitable introduction of driverless requirements to the F:SAE-A competition.

Figure 2 FSG regulation cones for autonomous events (adapted from [2]).

In this project, the feasibility of using reinforcement learning (RL) to allow local path following of a cone-marked track is investigated. Discussion begins with a brief overview of relevant literature before setting specific research goals. The methods used for development in simulation are then presented, before a comprehensive discussion of performance in both simulation and the physical world.

2. Literature Review

The literature review presented here first provides a brief history of autonomous vehicle development in research and industry. It then moves to look at reinforcement learning (RL) in a general sense and how these techniques have been applied to autonomous vehicles. Finally, autonomous vehicles in the context of the F:SAE driverless competition are evaluated, with a particular focus on the use of RL.

2.1 Autonomous Vehicles in Research and Industry

In March 2004, the first Defense Advanced Research Projects Agency (DARPA) Grand Challenge was held. This Challenge saw teams attempt to completely autonomously travel 225 kilometers from Barstow, California to Primm, Nevada (USA) with only a list of waypoints provided two hours prior to the event [3]. Of the 15 teams that raced, none made it more than 5% of the entire course [3]. In October 2005, a second Challenge was held. In this race, 23 teams competed and 5 finished. Many of the teams that competed, including the winning team from Stanford with their car STANLEY, published extensive documentation on their vehicle's design [3–5].

Since the DARPA challenge, many commercial vehicles have started to utilise some autonomous driving features. Some of these vehicles include the BMW750i xDrive (2015), Infiniti Q50S (2015) and Volvo XC90 (2020) which all include autonomous freeway driving, semi-autonomous parking and semi-autonomous braking functionality [6]. Perhaps the most advanced commercially available semi-autonomous vehicle at the moment is the Tesla Model S (2015), which includes all of the previously listed functions in addition to autonomous lane changing.

2.1.1 Autonomous Vehicle Systems Breakdown

As well as dividing vehicles by their autonomous capabilities, the various technologies required to get an autonomous vehicle to function can be broadly divided into three, well-known stages: see, think and act. 'See' covers vehicle perception from sensor characteristics, through to filtering, sensor fusion, object detection and classification. 'Think' covers mapping of the surrounding environment, localisation within this environment and path planning, while 'act' covers vehicle state estimation and actuator control; [7] provides an in-depth overview of many research papers covering all of these aspects. The focus of this research is on later half of the 'think' and the first part of the 'act' stages (see Figure 3).

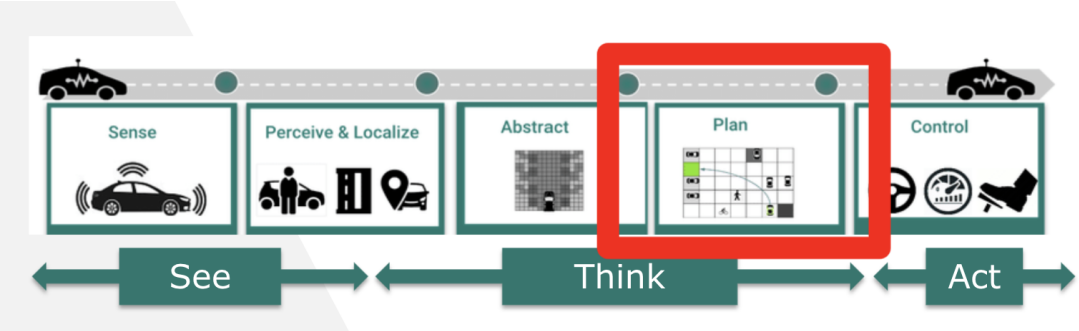

Figure 3 Autonomous vehicle modules and research focus (adapted from [8]).

2.2 Reinforcement Learning

Reinforcement learning is one of the three fundamental machine learning (ML) approaches (alongside supervised and unsupervised learning) and has been of research interest since the late 20th century [9]. It varies from the other approaches in that the algorithm autonomously collects data from an environment to train itself, rather than being provided it directly.

RL involves two distinct stages: *training* and *inference*. During the training stage, the *agent* performs *actions* on an *environment* and receives *observations* and *rewards* (determined by a *reward function*) from said environment (see Figure 4). The agent learns a *policy* to select actions that maximise the received reward, based on iterative trails it performs in an environment.

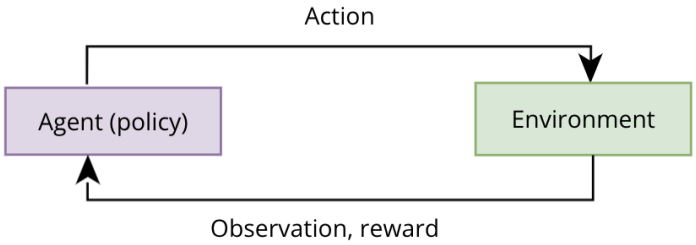

Figure 4 Agent-environment relationship in RL.

Trials are defined in two levels: *time-step* - a specific length of time for which a selected action is used and before another action is selected, and *episode* - a group of many time-steps before an environment must be reset (due to task completion, failure or timeout). Policy parameters are generally updated using an optimisation algorithm (such as gradient descent) after a certain number of episodes.

During the inference stage, the agent simply uses the trained policy to perform actions. The training stage must be performed at least once prior to the inference stage, which can be run an infinite number of times using the same trained model.

2.2.1 Common Reinforcement Learning Algorithm Classifications

Figure 5 summarises some of the many RL algorithms and categorisations discussed in [10]. This source, and [9] present two main methods of determining the optimal policy: *value-based*, e.g. *Q-learning* (works to maximise a value based on the reward e.g. a Q-value) and *policy-based* or *policy optimisation* (works to maximise the reward directly). A third method that combines these is known as *actor-critic*, and is discussed in detail in [11]. In actor-critic structures, the actor is policy-based while the critic is value-based. The actor is used to generate actions while the critic evaluates these actions. The critic is trained to determine how close the actor's action is to producing the optimal (Q) value, while the actor is trained to increase the value it receives from the critic. The networks are trained in parallel and can be thought of as working as adversaries.

Another important classification is *model-based* (requires a mathematical environment model) vs *model-free* (environmental model determined by trial and error). A further division is *on-policy* (collects observations using the model being trained) vs *off-policy* (collects observations using behaviours other than those generated by the model being trained) [12]. Inverse reinforcement learning (IRL) is also discussed in detail in [13]. IRL methods don't require defined reward functions like RL methods, but instead determine a reward function from demonstrated 'expert' actions. Imitation learning algorithms on the other hand attempt to directly imitate expert actions without generating an intermediate reward [14].

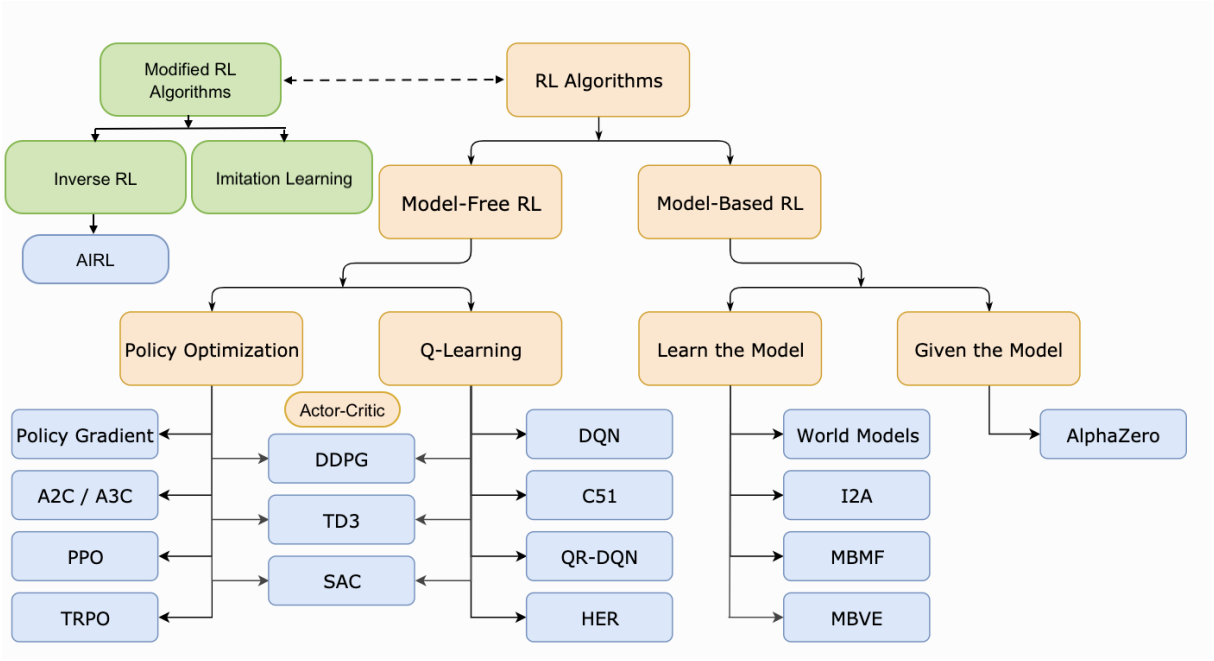

Figure 5 Taxonomy of some RL algorithms (adapted from [10]).

2.2.2 Deep Reinforcement Learning

In traditional RL, observations are related to actions and rewards in a tabular format. In more complex environments, the size of these tables quickly become limiting. Deep reinforcement learning (DRL) is a modification to RL where artificial neural networks (ANNs) are used to replace the relational tables. The first and most famous example of DRL is the out-performing of humans at some Atari games in 2013 [15]. This success of DRL has allowed RL algorithms to be used in more complex environments such as those navigated by autonomous vehicles (the subject of the following section).

2.2.3 Deep Reinforcement Learning and Autonomous Vehicles

The survey conducted in [16] focuses specifically on DRL methods for autonomous driving. It notes that DRL has been successfully applied to the problems of controller optimization, path planning, trajectory optimization, motion planning and dynamic path planning. It is discussed that DRL methods have the potential to be particularly effective in dynamic and unpredictable environments where traditional methods usually fail.

Several challenges and open questions related to DRL methods are identified in [16] including: validating performance (controller stability cannot be mathematically proven like in traditional control), covering the simulation-reality gap, sample efficiency (many episodes are required to train a DRL model), designing good reward functions and ensuring safety (DRL methods are somewhat 'black box' and may not be trained for every edge case). It is stressed that, like in [17], most DRL methods presented are only proven in simulation and haven't been tested in the more complex physical world.

2.2.4 Simulation to Reality Gap

By far the most common method to achieve a functioning DRL model in the physical world is to first train it in simulation [18]. Doing this somewhat addresses the sample efficiency and edge-case problems as it is much faster and cheaper to gather data in simulation. Unfortunately, this introduces many of its own problems as discussed in [18].

Most of these problems revolve around building a realistic simulation due to the difficulty of accurately modelling camera images and building dynamics models of all bodies in an environment.

Some approaches to addressing the simulation to reality gap are discussed in [18]. These include: *domain randomisation* (randomly altering parameters of the simulation environment such as sensor noise), *transfer learning* (training a model trained in a source environment in a new target environment) and *domain adaptation* (a sub-classification of transfer learning where the source is simulation and the target is the physical world). As insufficient literature has been presented where these methods are implemented, the transferability of DRL methods from simulation to the physical world is still an open question.

2.3 Autonomous Vehicles in F:SAE

Most publicly-available autonomous F:SAE papers focus on higher-level vehicle design, or design of a specific subsystem [19–22]. The most robust and complete research on the performance of an autonomous F:SAE vehicle belongs to the Academic Motorsports Club Zurich (AMZ). AMZ have published a series of four papers focused on autonomous perception in an F:SAE context: in 2018, [23] focused on early perception and state estimation; in 2019, [24] focused on cone detection using LiDAR and [25] focused on cone detection using computer vision; and in 2020, [26] gave a complete system overview and a summary of AMZ's autonomous research.

To our knowledge, the only paper that discusses the use of a RL-related technique in an F:SAE context is [27]. This paper describes the use of an imitation learning algorithm used with a modified PilotNet ANN for end-to-end autonomous driving. To train the model, data was gathered by human drivers in model F:SAE vehicles using the photo-realistic AirSim simulation software. This method is however limited in that the autonomous vehicle can never outperform the human drivers. Further, human drivers are required for data collection and parameters of the vision module cannot be altered independently of the DRL model.

3. Research Goals

The higher-level vision of this research is to create the foundation for developing an autonomous F:SAE car at the University of Auckland. The specific contributions to the field beyond this goal include: validating the performance of some RL algorithms in the physical world, overcoming the simulation-reality gap in this context and a discussion of the design of reward functions for autonomous racing purposes.

3.1 Objectives

Three objectives are defined for this project:

- Evaluate and compare the training and inference performance of selected RL algorithms on a single track in a representative F:SAE simulation.
- Evaluate how the the inference performance of the trained RL models varies across different tracks in simulation.
- Evaluate the inference performance of the trained RL models on a track in the physical world.

3.2 Methodology and Scope

Selected DRL algorithms are trained in simulation on the FSG 2019 trackdrive track. These algorithms will not cover the 'see' stage (see red box in Figure 3). Note that 'training performance' refers to the number of episodes to convergence.

Inference performance is tested on the training track, and the reversed training track, in simulation. Note that 'inference performance' refers to how far around the tracks the models can successfully navigate and how consistent their performance is in doing so. Inference performance is also tested on a physical kart and scaled tracks representing the original training track. Both transfer learning and domain adaptation are out of scope.

The scope includes changing the way RL algorithms interact with the F:SAE environment, not a fundamental alteration of the backend of the RL algorithms. This excludes ANN structure changes (such as number of hidden layers or activation functions) or learning rate changes from the scope. Further, only stereo cameras and steering angle sensors are considered in-depth and the vehicle speed is always held constant for simplicity. The total cost of the physical prototype developed is less than $300 NZD excluding the components available for loan from The University of Auckland.

4. Simulation Setup and Results

Using simulation allows RL models to be trained and tested without many hours of human driving data. This section first discusses the RL algorithms selected to train the driving models before giving an overview of the higher-level software structure used and a discussion of the simulation environment. The training results gathered from this environment are then presented and discussed, followed by inference results from simulation.

4.1 Simulation Setup

4.1.1 Selected RL Algorithms

Soft Actor Critic (SAC): See Figure 6 for a graphical representation of the functioning of SAC. Note that the actor and critic are modelled by separate, fully-connected ANNs.

SAC is a model-free, off-policy, actor-critic RL algorithm first proposed in [28]. Because SAC is off-policy, a behaviour generator is used to generate somewhat stochastic actions in an environment. The reward and observation returned by an environment are passed to the critic network in the agent. Outputs from the critic are then used to update ANN weights in the actor and critic using an optimiser (Adam optimiser in this case). The ANN parameters used in this project are the parameters that worked in the 'Hopper' and 'Inverted pendulum' environments in the original implementation (also true of the next algorithm) and can be seen in Appendix C. Occasionally, the actor is used to update the behaviour generator so observations more similar to the trained policy can be obtained.

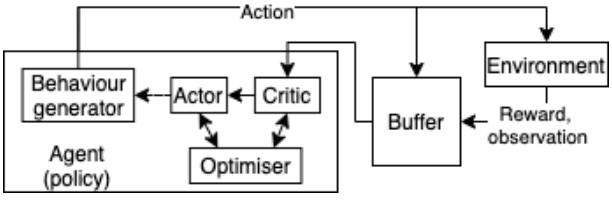

Figure 6 SAC component summary.

The advantage of SAC over other RL algorithms is its robustness to imperfect hyperparameters and observations. This is due to the "maximum entropy" approach where the behaviour generator generates many random actions to better explore the environment [28].

Adversarial Inverse Reinforcement Learning (AIRL): AIRL is a model-free, inverse reinforcement learning algorithm (IRL) first presented in [29]. This means that rather than exploring the environment and filling the buffer with observations on its own, it uses an "expert buffer". This is a collection of observations generated by an "expert" (an agent that performs well in the environment) that must be created prior to training the algorithm.

Figure 7 shows that the structure of the AIRL system used is very similar to SAC shown in Figure 6. Apart from the expert buffer, the biggest change is that no reward is required from the environment. The reward is instead replaced by a "discriminator" ANN that attempts to tell the difference between an expert observation and the real observation generated by the same action. The advantage of this is that no reward function has to be defined which can be difficult in some situations. However, AIRL, like all other IRL algorithms, must be trained by an expert which is not possible in all cases. In this project, the *expert* in simulation is simply a PID controller that moves the vehicle towards a hard-coded centre line of the training track (called *pure pursuit*).

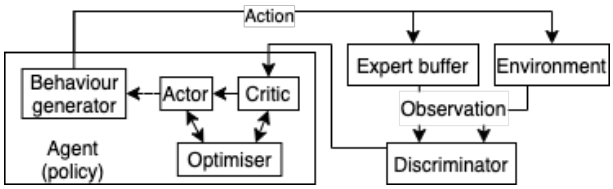

Figure 7 AIRL component summary.

4.1.2 Higher-Level Overview

The software project is based on many sets of custom open-source software freely available online. All code is implemented using Python 2.7 and C++, tied together through the ROS (Melodic) framework and run on Ubuntu 18.04. The source code for the entire project (including the simulator, RL algorithms and custom packages) can be found at: https://github.com/HarveyMerton/uoa_fsd. For a complete list of repositories used, see Appendix D.

Figure 8 is a higher-level dependency graph of the main modules used in this project (see Appendix E for how this relates to the code implementation). Note that 'simulator backend' and 'RL backend' represent the complex software backends required to run the simulator and learning algorithms respectively. Only minor modifications are made to these backends mostly to allow transfer of information for training.

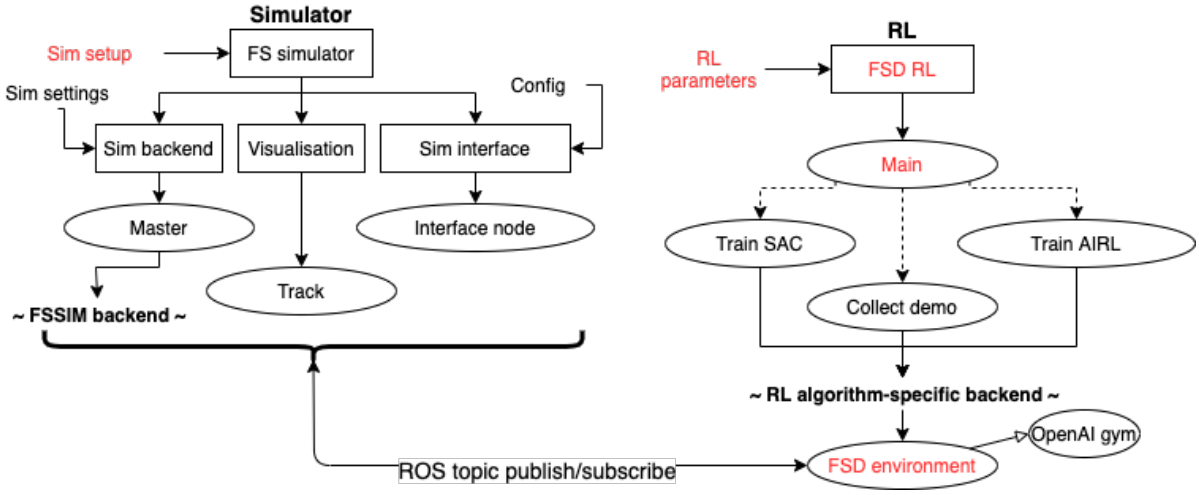

Figure 8 Higher level summary dependency graph. Key altered modules in red.

Figure 8 shows that the project is run in two parallel processes: simulator (discussed in Section 4.1.3) and RL (algorithms discussed in Section 4.1.1). Note the dotted lines

in the RL section means that only one of the three submodules shown will be executed depending on what learning algorithm is to be used. Note also that 'FSD environment' is the module where most of the novel work that connects the simulator and RL algorithms is implemented. Its implementation is based on the OpenAI Gym toolkit in ROS [30].

4.1.3 FS Simulator Environment

The FS simulator was developed by the AMZ Driverless team to model a realistic Formula Student vehicle in ROS (see Figure 9 for a sample visualisation). It is based on a vehicle model discretized through a forward Euler approximation [26] and has the following characteristics (relevant parameters of the simulated vehicles and tracks can be found in Appendix F):

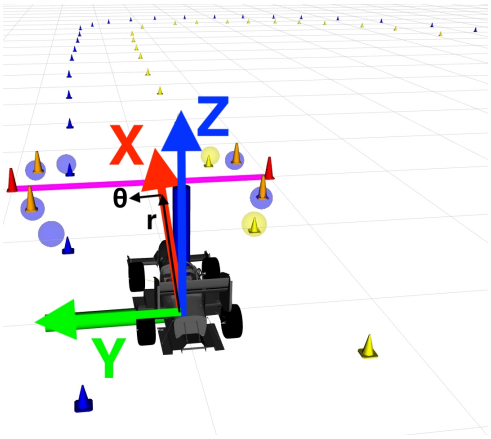

Figure 9 FS simulator visualisation - RVIZ.

- **Degrees of freedom** - chassis has 3 (X, Y, yaw), front wheels have 2 relative to chassis (Y and Z rotation).
- **Control inputs** - steering angle and throttle position normalised from -1 to 1.
- **Sensors** - "stereo camera" and "lidar". These sensors are represented as independent spherical regions around the vehicle within which cone positions can be "seen" (note that real representative camera and lidar sensors are not simulated). Both have adjustable range, Gaussian noise on observed cone pose, and cone observation probability.
- **Kinematics modelling** - track width, wheelbase and steering centre included, no suspension modelling.
- **Dynamics modelling** - point mass with centre of mass position and front/rear weight balance, yaw inertia, tyre model (Pacejka's magic formula), basic aerodynamics model (whole vehicle lift and drag co-efficients) and basic torque vectoring.

Co-ordinate systems (CS): Two are defined in Figure 9: {X, Y, Z} - origin is attached to the base of the chassis in the centre of the driver's seat and X is aligned with the chassis, {r, θ} (used for cone position noise) - same origin as {X, Y, Z} and $\theta = 0$ is aligned with the X axis. A third, static, global CS {x, y, z} is defined as the {X, Y, Z} CS before the vehicle moves after an environment reset. The {X, Y, Z} CS is pictured in the pose defining the {x, y, z} CS in Figure 9.

Environment setup: Two important environment descriptors required in all RL environments are *action space* and *observation space*. These descriptors allow learning functions to determine the number and form (e.g. integer or decimal) of the actions to be generated and in what form an observation of the environment should be expected. These descriptors, as well as other key environment variables, are outlined below:

- **Action space** - A single variable to represent a selected normalised steering angle. The value is continuous and lies in the range -1 to 1 (corresponds to ± 45 deg). Limits are applied to reduce this to -0.4 to 0.4 (corresponds to ± 18 deg) as this is the full steering range used by the expert.

- **Observation space** - A 6x3 matrix where each row corresponds to a single observed cone: 3 blue and 3 yellow in total (as 3 points are required to define a curve). The columns correspond to the X and Y co-ordinates as well as a *colour identifier* (colour ID: 1 for blue, -1 for yellow) of each cone. Note that the (X,Y) co-ordinates are defined in co-ordinate system {X, Y, Z} and so are relative to the moving vehicle.

- **Time-step size** - The ROS time that a 'step' in the environment runs for before a new action (steering angle) is selected. 0.1 s is selected; the effect of selecting different time-step sizes on inference performance is discussed in Section 4.3.

- **Speed** - The vehicle speed is set at a constant 4 m/s (14.4 km/h) as this is the default speed provided in the simulator and a reasonable speed for a driverless vehicles's first lap of an unknown track. The effect of varying the speed on inference performance is discussed in Section 4.3.

- **Rate limit** - A hard limit placed on the rate at which the steering angle can be changed. This prevents the RL algorithms causing physically impossible instantaneous angle changes.

- **Camera range** - Set at 10 m as the simulated camera can mostly capture at least 3 blue and 3 yellow cones at this range. The FSD environment module adds additional cones at the origin i.e. the center of the vehicle (see Figure 9) and cuts out further cones to ensure that 3 blue and 3 yellow cones are always provided to the learning algorithm (same number always required by the ANN in the RL backend). Additional cones are placed at the origin so the RL algorithms can learn to clearly identify these false cones (cones should never naturally appear at the origin) and not act on them.

- **Uncertainty in cone position measurement** - Gaussian noise is used in the FS simulator to model uncertainty in cone positions. This is modelled by the default values of $\mu_{r,\theta} = 0.0$, $\sigma_r = 0.2$ and $\sigma_\theta = 0.007$ (polar co-ordinates relative to the black arrows in Figure 9 where μ is mean and σ is standard deviation). These values are representative of the noise observed by AMZ using their cone detection system [26].

Track:

Figure 10 shows the track defined for training and inference testing in simulation. This track is representative of the 2019 FSG trackdrive track and thus contains a variety of key features expected on an F:SAE competition track. Some of these key features are marked in Figure 10 but in general, note the combination of straights and right and left hand turns of various radii and lengths (arrows distort dimensions slightly). The track is approximately 400 m in total length.

The arrows shown in Figure 10 represent the direction of vehicle travel. To allow inference testing on an environment different to the training environment, but still representative of a general F:SAE driverless track, the vehicle travel direction (and corresponding coloured cone sides), are simply swapped. This inverse track is referred to as the 'inverse FSG track', whereas the original track is the 'FSG track'.

On this track, and all others used in testing, one episode is defined as the time from when the vehicle starts in the position shown in Figure 10 until all four vehicle wheels leave the cone-marked track, or a full lap is completed (vehicle crosses pink finish line for the second time).

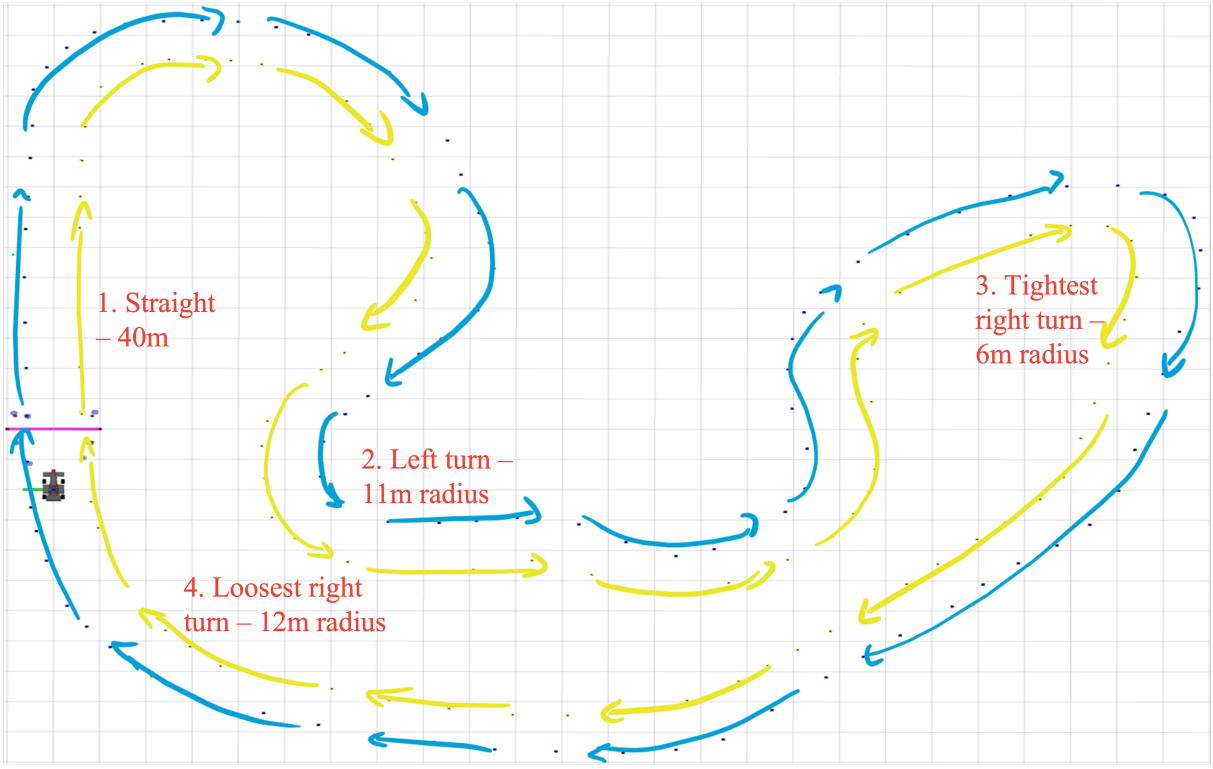

Figure 10 FSG 2019 full-scale trackdrive track in simulation.

Reward functions: These custom-defined functions are used to generate the reward received from the environment (at the end of every time-step) during the training process (see Section 2.2). Recall that these functions are used to train SAC but not AIRL (see Section 4.1.1).

Three 'fundamental' reward functions are defined in the FSD environment and have some common symbols: i represents time-step i, α_1 through α_7 are tunable scaling parameters, and MAX is a cap placed on the reward able to be received at any one time-step (prevents reward explosion when dividing by small numbers). $done_i$ is a binary variable which equals 1 if the episode has ended and 0 if not. Note that '!' is a logical 'NOT'.

Function 1

$$Reward_i = min(!done_i \times (\alpha_1 + \frac{\alpha_2}{|ang_i - ang_{i-1}|}),\ MAX) \tag{1}$$

Where ang_i is the steering angle at time-step i. Setting $\alpha_2 = 0$ in reward Function 1 gives the simplest reward function possible; a reward of α_1 is given for every time-step that the vehicle remains between the cones, otherwise a reward of 0 is given. As a constant throttle value is sent to the simulator at all time-steps, the number of time-steps the vehicle achieves in an episode is roughly proportional to the distance it has travelled around the track. Thus, the cumulative reward received for a full episode is roughly proportional to the distance travelled around the track.

Setting α_2 to a non-zero value in reward Function 1 results in a higher reward given a smaller steering angle change (relative to the last time-step). This modification is intended to increase the 'smoothness' of travel around the track.

Function 2

$$Reward_i = done_i \times \alpha_3 + min(!done_i \times \frac{\alpha_4}{\sqrt{(tx_i - px_i)^2 + (ty_i - py_i)^2}},\ MAX) \tag{2}$$

Where tx_i, ty_i, px_i and py_i are the target (x, y) and current (x, y) positions in the global CS at time-step i respectively. The target position is the midpoint between the furthest pair of blue and yellow cones seen in time-step $i-1$ (see Figure 11); reward is given by how close the vehicle is to this point in the next time step. Note that α_3 can be used to give a negative reward when the vehicle exits the track.

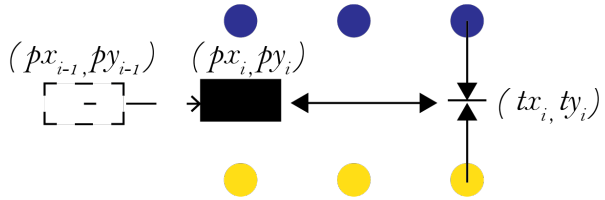

Figure 11 Reward function 2 visualisation.

Function 3

$$Reward_i = done_i \times \alpha_5 + min(!done_i \times (!target_i \times \alpha_6 + \frac{target_i \times \alpha_7}{c_i - c_t}),\ MAX) \quad (3)$$

Where c_i and c_t are the total time-step counts (since episode start) at time-step i and t (time-step when the target was set) respectively. $target_i$ is a binary variable which equals 1 if the vehicle has reached a predefined target and 0 if not. The predefined target is the midpoint between the furthest pair of blue and yellow cones observed at time-step c_t (similar to Figure 11). Once the target is reached in time-step c_i, a new target is generated and c_t is set to c_i.

The target position in this function is similar to Function 2 except the same target is kept until it is reached, rather than moving the target each time a new set of cones can be observed. This allows the reward to be associated with time, thus linking the reward to the ultimate objective of minimising lap time. Due to time restrictions, this function was unable to be tested in this project. Recommendations for its future use are provided in Section 7.

4.2 Training Results and Discussion

Training results are summarised in Table 1. Note that the 'IDs' assigned to each row (training simulation) are not in order. This is because training simulations were performed in parallel on multiple machines and only key results are presented.

Table 1 also reports "angle limit", "rate" and "colour ID". These parameters are discussed in detail in Section 4.1.3 and represent if a $\pm 18°$ steering angle limit, steering angle rate limit or cone colour identifiers are applied. "Rollout" length also referenced refers to the number of steps ahead the AIRL algorithm is looking at when optimising the selected steering angle.

Convergence definition: The algorithms are considered converged when 5 consecutive full laps have been completed during training. Table 1 reports 'episodes to convergence' in three stages: when 1, 2 and 5 consecutive full laps have been completed (one per episode as episodes end after a complete lap). Note that the episode reported is the first episode in the sequence of episodes where a full lap is performed.

ID	What testing	Episodes to convergence					
	SAC	1	2	5	Angle limit	Rate ($^\circ/s$)	Colour ID
	Function 1						
S1b	$\alpha_1 = 1$, $\alpha_2 = 0$	271	314	504	N	N	N
S14	$\alpha_1 = 1$, $\alpha_2 = 0$	326	328	362	Y	N	N
S2b	$\alpha_1 = 100$, $\alpha_2 = 0$			176	N	N	N
S3b	$\alpha_1 = 1$, $\alpha_2 = 1$	510		528	Y	N	N
	Function 2						
S4b	$\alpha_3 = -10$, $\alpha_4 = 10$	751	1303	2278*	N	N	N
	Steering angle limits	($\alpha_1 = 1$, $\alpha_2 = 0$)					
S1b	$\pm 45^\circ$	271	314	504	N	N	N
S6	$\pm 18^\circ$	250		352	Y	N	N
	Colour identifiers	($\alpha_1 = 1$, $\alpha_2 = 0$)					
S6	Without	250		352	Y	N	N
S15	With	235	239	351	Y	N	Y
	Rate limit (deg/s)	($\alpha_1 = 1$, $\alpha_2 = 0$)					
S15	None	235	239	351	Y	N	Y
S12b	90	521	681	1942*	Y	90	Y
S18	112.5	456	525	735	Y	112.5	Y
S17	135		477	615	Y	135	Y
S16	180	523	578	624	Y	180	Y
	Cone pose uncertainty	($\alpha_1 = 1$, $\alpha_2 = 0$)					
S18	Default	456	525	735	Y	112.5	Y
S21	None		376	443	Y	112.5	Y
	Fail condition	($\alpha_1 = 1$, $\alpha_2 = 0$)					
S18	All wheels out	456	525	735	Y	112.5	Y
S19	One wheel out			6556*	Y	112.5	Y
	AIRL	1	2	5	Rollout	Rate ($^\circ/s$)	Colour ID
	Rollout length (steps)						
A2	50 K			3354*	50 K	N	Y
A4	25 K	1159	1272	1356	25 K	N	Y
A5	12.5 K	534	664	752	12.5 K	N	Y
A3b	6125	685	834	1277*	6125	N	Y
A10	1000	98	117	121	1000	N	Y
A12	500	221		316	500	N	Y
A9	100	11	13	69**	100	N	Y
	Cone pose uncertainty						
A10	Default	98	117	121	1000	N	Y
A11	None	541	896	1700	1000	N	Y
	Rate limit (deg/s)						
A10	None	98	117	121	1000	N	Y
A13	112.5	636	1329	1386	1000	112.5	Y
	* = Max episode (not converged), ** = unstable, Y = yes, N = no						

Table 1 Training results summary.

The definition of convergence is drawn from repeatable inference performance. Figure 12 shows the percentage of the FSG track completed (over 10 inference trials) by SAC models saved at various stages of a training run. The results show that variability in inference performance generally decreases as the number of consecutive laps completed increases (inter-quartile range (IQR) decreases from 40% after one lap to 25% after 5 laps). The results also show that the lap number within the streak at which the model is saved is important: a model saved at lap 2/5 has an IQR = 0 whereas one saved at 5/5 has IQR = 25%. This means that a model must be saved in the first few of 5 consecutive laps to be considered converged.

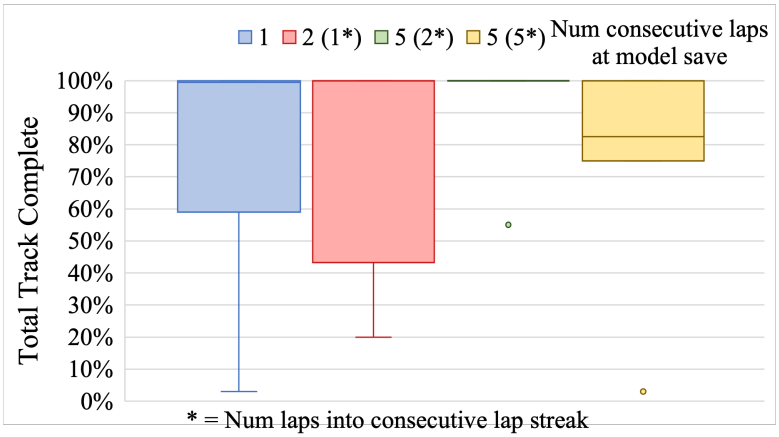

Figure 12 Repeatability of SAC models after different numbers of consecutive full laps in training.

4.2.1 SAC

Figure 13 shows a typical training graph for the SAC algorithm in the FSD environment. Full laps can be seen where the reward value plateaus at around 900 in this case.

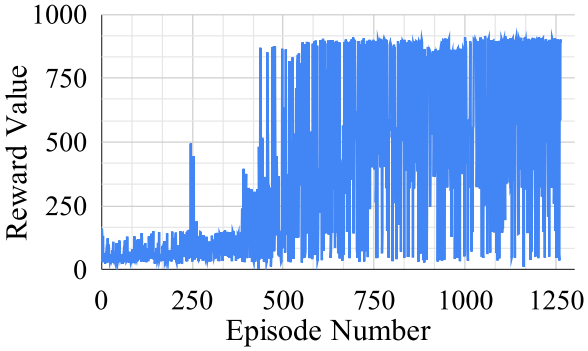

Figure 13 Final SAC training graph (S18).

Reward functions: Comparing results S14 and S3b in Table 1, it can be seen that adding a non-zero α_2 into reward function 1 appears to increase overall convergence time from 362 to 528 episodes. This is expected as more complex reward functions are generally more complex for RL algorithms to decipher. However, qualitative analysis of vehicle trajectory shows no improvement in “smoothness" as desired. This is due to the backend ANN having no “memory" of the previous steering angle so it is unable to learn that a small change from the previous angle gives a larger reward. For this reason, $\alpha_2 = 0$ going forward.

Comparing S1b with S2b it is seen that increasing the magnitude of α_1 from 1 to 100 drastically reduces the number of episodes required before convergence. As this was discovered after most other training tests, $\alpha_1 = 1$ for most of the remaining tests.

Observing S4b, it can be seen that the only test of reward function 2 did not converge. It was later discovered that this may have been due to an implementation error. However, as reward function 1 had proven functionality, it is selected as the base function going forward.

Steering angle limits: Adding a steering angle limit reduces convergence time from 504 to 352 episodes (compare S1b with S6). For this reason, the steering angle limit is applied going forward.

Cone identifiers: Adding cone identifiers increases convergence time from 352 to 488 episodes. However, qualitative inference performance suggests a much straighter trajectory when identifiers are included. As the impact on training performance isn't overwhelming, cone identifiers are included going forward.

Rate limit: Applying a low rate limit of $90°/s$ prevents the algorithm from converging (compare S12b and S15). Testing a high rate limit of $180°/s$ proves that the algorithm is able to converge with a rate limit applied. Performing a binary search between these values, it is found that a rate limit of $112.5°/s$ still allows the algorithm to converge, but also reflects a physically possible rate of steering angle change. Unfortunately, adding this $112.5°/s$ rate limit significantly lengthens the training process (351 to 735 episodes). However, without it, the model will not be physically realisable. Thus, the $112.5°/s$ rate limit is applied going forward.

Cone pose uncertainty: Comparing S18 with S21, removing the uncertainty in cone positions during training significantly reduces training time (735 to 443 episodes). However, to follow the literature concerning domain randomisation, cone uncertainty in training is retained to improve trained model robustness. Confirmation of the actual effect of cone pose uncertainty on robustness is needed in the future.

Failure condition: Hitting cones results in loss of points in the FSG competition. In an attempt to train a model that hits no cones, S19 was run where one wheel outside of the track constitutes failure. As the vehicle has so little space to initially explore the environment with this condition, it fails to converge after 6,556 episodes. Compare this with the 735 episodes for convergence in S18 where all four wheels must be outside of the track for failure. As the FSG rules use this second failure condition [1], it is acceptable to continue with S18. Thus, *S18 is selected as the SAC model to take forwards to inference testing*.

4.2.2 AIRL

Figure 14 shows a typical training graph for the AIRL algorithm in the FSD environment. As explained in the AIRL discussion of Section 4.1.1, an expert is required to train the algorithm and so does not use the displayed reward values. However, full laps can be still be seen where the reward value plateaus at around 1250 in this case. The results shown for AIRL in Table 1 are achieved using data collected by a pure pursuit expert on the FSG track with the default uncertainty on cone positions.

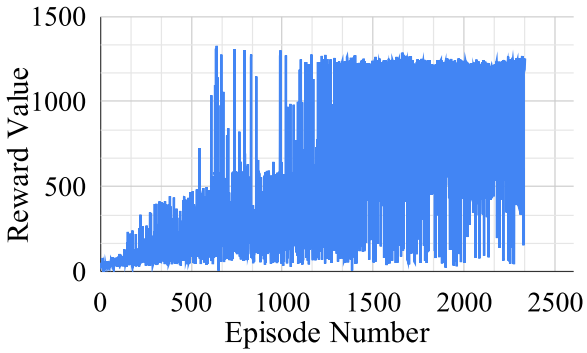

Figure 14 Final AIRL training graph (A13).

Rollout length: The rollout length was initially set at 50 K steps which worked in the Hopper environment (see Section 4.1.1). As convergence did not occur after 3,354 episodes, a binary search was performed with 0 as the lower bound. As the rollout length was decreased, episodes to convergence also decreased to reach 121 steps with a rollout length of 1,000 (A10). This is with the exception of the length 6,125 steps where convergence

was not reached after 1,277 episodes. Although A9 shows extremely fast convergence with 69 episodes, the performance quickly dropped off and no full laps were completed until episode 826, when convergence was again reached. This suggested unstable convergence and so 1,000 steps was selected as a fast converging and stable rollout length with which to proceed.

Cone pose uncertainty: In the opposite manor to SAC, removing cone pose uncertainty actually increased convergence time from 121 episodes in A10 to 1,700 in A11. This is likely due to the expert data being collected with the cone pose uncertainty present. Again, further testing of this hypothesis is required, but for a similar reason to SAC (using domain randomisation), cone pose uncertainty remains at the default level going forward.

Rate limit: Applying the $112.5°/s$ rate limit drastically increased the convergence time from 121 to 1386 episodes (11.5 times increase as compared to the 2 times increase seen in SAC). A slightly larger rate limit could have be investigated to attempt to reduce convergence time, but $112.5°/s$ was selected to maintain comparability with SAC. Thus, for comparability and to allow physical implementation, the $112.5°/s$ rate limit was carried forward. As such, *A13 is selected as the SAC model to take forwards to inference testing*.

4.3 Inference Results and Discussion

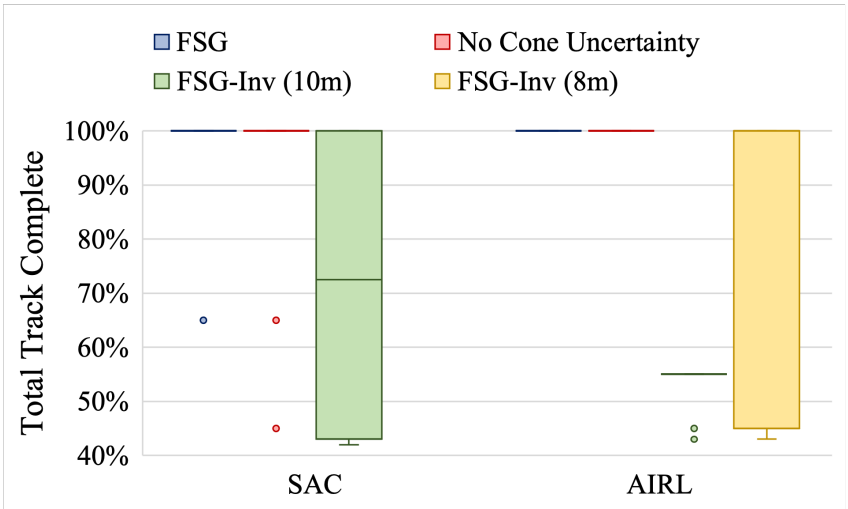

Figure 15 SAC vs AIRL simulation inference performance.

The models used in this section are those trained by the S18 (SAC) and A10 (AIRL) training simulations.

4.3.1 FSG Track

Figure 15 compares the inference performance of SAC and AIRL-trained models on the FSG and inverse FSG (FSG-Inv) tracks. The performance is measured by percentage of the total track completed over 10 inference runs. As seen in Figure 15, both SAC and AIRL-trained models achieve a median of 100% track completion on the FSG track. Similar performance is obtained when cone pose uncertainty is removed on the inference tracks suggesting that domain randomisation may have helped increase robustness to changing cone pose uncertainty. This would need to be verified by comparing the inference results of a model trained without cone pose uncertainty.

A closer look at Figure 15 shows that SAC has 1 trial not completing the full FSG track and two not completing the track with no cone uncertainty. This is unexpected as a

converged model would be expected to see 100% completion on the track it was trained on. It is likely that this variation, and other variation seen in the following results, is partially due to the non-blocking nature of the independent ROS topics through which environment data is captured (delays in these topics will cause the models to act on old data). AIRL appears less vulnerable to this variation (all 10 trails achieve 100% track completion) as its trajectory is generally closer to the centre of the track and so poor decisions are easier to recover from.

4.3.2 Inverse FSG Track

Figure 15 shows that the performance of both SAC and AIRL-trained models is worse on the inverse FSG track than the FSG track. The SAC model shows a median of around 72% track completion. This is misleading as the results are clustered in two groups: around 43% - 45% and 100%. The 43% - 45% failure points correspond to the tightest right turn in the original FSG track (see feature 3 in Figure 10). As this corner has a smaller radius than any left turn in the original FSG track, it constitutes a feature of the inverse FSG outside of the original training bounds. This shows that SAC is not able to generalise well to a corner this far out of its original training data. The 100% completion clustering of 5 trials however shows that for the trials where the vehicle is able to make it past this obstacle, the rest of the track can be easily completed; SAC can generalise to overcome features similar to its original training data.

In contrast to the SAC-trained model, the AIRL-trained model shows a median and maximum track completion of 55% around the inverse FSG track. This corresponds to the point where two parts of the track are closest (see corner between features 2 and 3 in Figure 10). Using the default 10 m camera range here results in cones on another leg of the track being observed. These throw off the AIRL-trained model and cause it to fail. The fact that the SAC-trained model doesn't fail at this point with the same camera range suggests that the AIRL-trained model is more sensitive to "random" cone observations than SAC. Reducing the camera range to 8 m prevents this phenomenon and so AIRL sees a similar clustering of results to SAC (4 fail at 43% and 6 complete 100%). The fact that AIRL is able to have the camera range reduced and still perform well shows some robustness to camera range changes.

4.3.3 Trajectory and Smoothness

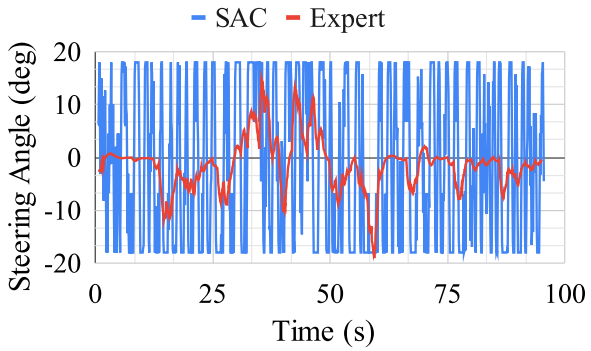

Figure 16 SAC vs expert inference on FSG track.

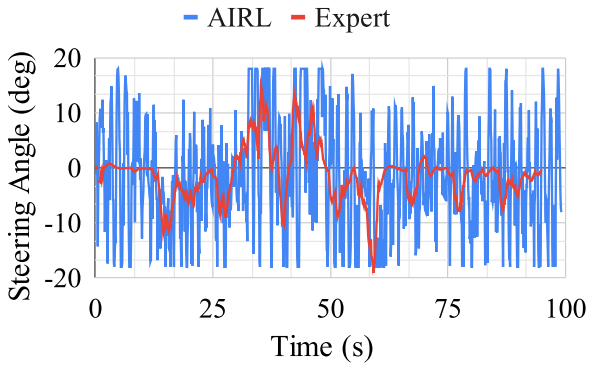

Figure 17 AIRL vs expert inference on FSG track.

Figures 16 and 17 compare the steering angle selected by the expert to the RL models during a lap around the FSG track. It is clear that the AIRL-trained model is more similar to the expert trajectory which corresponds to less zig-zagging around the track. The mean

derivative steering angle from these graphs presented in Table 2 show that both RL models are significantly “noisier"/less smooth than the expert: expert has mean derivative $4.4°/s^2$ while SAC and AIRL have $57.3°/s^2$ and $54.7°/s^2$ respectively. The second column in Table 2 shows a similar trend on the FSG inverse track. It is worthy of note that the the SAC-trained model has a much smaller deviation in mean derivative from the FSG to FSG inverse track ($0.3°/s^2$ vs $2.2°/s^2$) which suggests marginally more consistent performance across different tracks. Links to videos of the vehicle generating these results can be found in Appendix G.

	Mean steering angle derivative (deg/s^2)	
Model	FSG	Inverse FSG
Expert	4.4	N/A
SAC	57.3	57.6
AIRL	54.7	56.9

Table 2 Steering smoothness of models across different tracks.

4.3.4 Speed and Time-step

		Speed (m/s)							
		SAC				AIRL			
		2	4	5	6	2	4	6	7
Speed x timestep	0.2	0.1	0.05	0.04	0.033	0.1	0.05	0.033	0.03
	0.4	0.2	0.1	0.08	0.067	0.2	0.1	0.067	0.057
	0.8	0.4	0.2	0.16	0.13	0.4	0.2	0.13	0.11
Cell contents:		1 (2)	0.3 (1.2)	0.2 (1)		3 (6)	0.55 (2.2)	0.167 (1)	
timestep in sec		1.5 (3)	0.35 (1.4)	0.25 (1.25)		3.5 (7)	0.6 (2.4)	0.2 (1.2)	
(speed x timestep)		2 (4)	0.4 (1.6)			4 (8)	0.65 (2.6)		

Key	Mean track completion (%)					
	100	90-99	80-89	70-79	60-69	<60

Table 3 Inference performance with speed and time-step variations.

Table 3 summarises the inference performance of the RL models across five trials at different speed and time-step combinations. The hypothesis was that the performance would depend on the speed $\times$ time-step product (see the first three rows of Table 3). However, it was found that this product didn’t carry much significance and rather there was simply a maximum time-step at each speed after which performance rapidly dropped off. For both SAC and AIRL, this maximum time-step reduced as speed increased. For SAC this was from 1 s to 0.3 s and 0.16 s at 2, 4 and 5 m/s respectively and for AIRL: from 3 s to 0.55 s and 0.167 s at 2, 4 and 6 m/s respectively. Note that the AIRL model is capable of performing to a higher time-step at each speed and to a higher maximum speed (6 m/s vs 5 m/s) than the SAC model when both are trained at a speed of 4 m/s and a time-step of 0.1 s.

5. Physical World Setup and Results

5.1 Hardware Setup

Figure 18 shows the setup of the hardware system used for physical testing (see Appendix I for more detailed images). As no functioning F:SAE vehicles are available, the system is built on a kids’ go-kart with Ackermann steering, roughly half the size of an F:SAE vehicle (see Appendix C for a comparison of vehicle parameters). PVC pipe and a 3D printed platform is used to hold the stereo camera in a position comparable to being mounted on an F:SAE roll hoop; a popular position for other F:SAE teams [26].

Figure 18 Physical Testing Kart.

A magnetic rotary position sensor mounted on the steering wheel is used to observe the steering angle (see Appendix L for setup).

During a test, a driver sits in the kart and is pushed around a test track (see Figure 21 for an example track) at roughly 4 m/s. Note that this speed is only estimated (not measured). As the algorithms have proven to function well between 2 - 5 m/s on a full scale vehicle (1 - 2.5 m/s at this half scale) with a step size of 0.1 s, exact speed sensing is not required to keep the estimated speed within this (large) range.

While being pushed, the driver simply turns the steering wheel in the direction and by the (degree) amount shown on a laptop's screen sitting on their lap (see Figure 19 for the driver's display), attempting to follow the commands as closely as possible. A blanket is placed over the front of the kart during testing to ensure the driver navigates according to this angle only and no other visual cues. The specifications of the laptop used for this testing can be found in Appendix K.

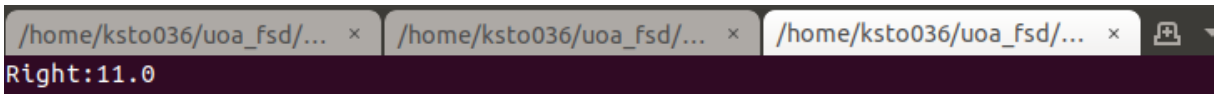

Figure 19 Driver display during physical testing.

5.1.1 Tracks

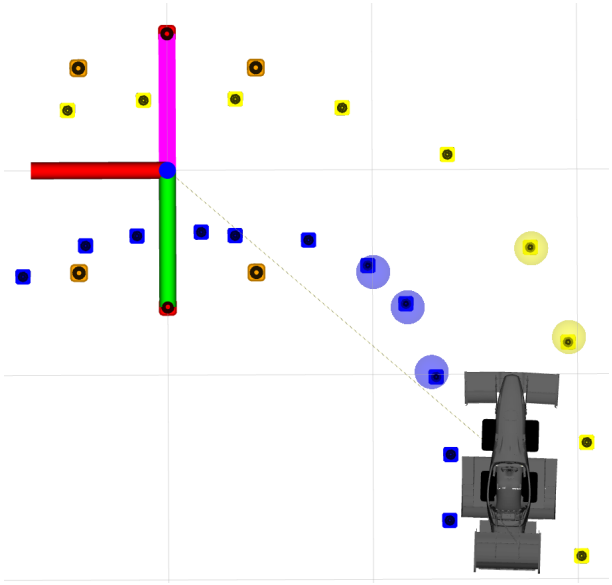

Figure 20 Simulated left turn track setup.

Figure 21 Physical left turn track setup.

Tracks in the physical world are delineated with ArUco markers simulating blue and yellow "cones" (see Appendix J for marker detection examples). Due to the poor performance of the marker detection system outdoors, track segments were set up for testing indoors. Figure 21 shows one such track.

Four tracks are designed, each representing one of the features marked in Figure 10: a straight track (6 m), left turn (90° of a 5.5 m radius), tight right turn (130° of a 3 m radius) and loose right turn (60° of a 6 m radius). As the radii of the corners and width of the tracks are approximately halved to match the kart's scale, a half scale simulation environment is also built (compare Figures 20 and 21 and see Appendix H for other tracks). This allows testing of the scaling performance of the RL models independent of the physical world transferability.

5.2 Inference Results and Discussion

Figure 22 compares the track completion across the four simulated (scaled) and physical tracks for both the SAC and AIRL algorithms. The results are drawn from 10 repetitions on each track. Clearly, models trained by both algorithms perform very well in both the scaled and physical environments with a median completion rate of 100% across all tracks. The AIRL-trained model proved to be slightly more repeatable when scaling down in simulation; 100% track completion in all trials vs 92.5% of trials achieving full completion from SAC. Similarly, the AIRL-trained model performed slightly better when transferred to the physical world with 95% of all trials achieving full completion compared to a total completion rate of 87.5% from the SAC-trained model.

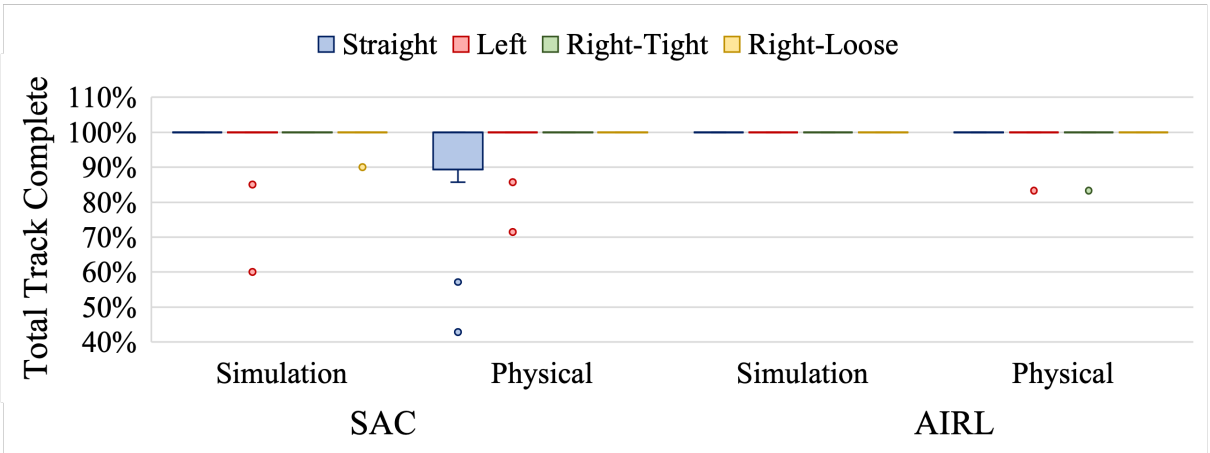

Figure 22 SAC vs AIRL inference performance on (scaled) simulated and physical tracks.

Figures 23 and 24 show the difference in the actual kart steering angle and the angle desired by the RL models during one trial on the straight tracks. Although the error was moderate (mean absolute error of 7.5° for SAC and 9.9° for AIRL), both models remained robust to this. This error was unavoidable due to no available drive-by-wire testing system.

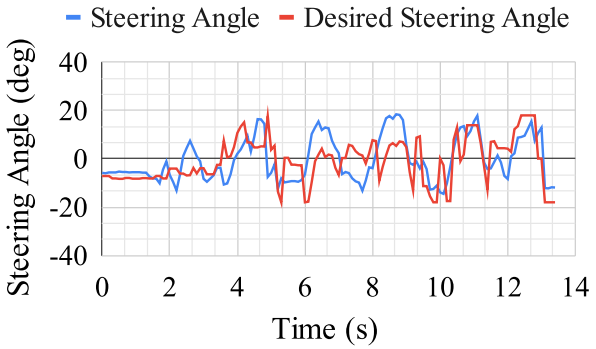

Figure 23 SAC inference on straight physical track.

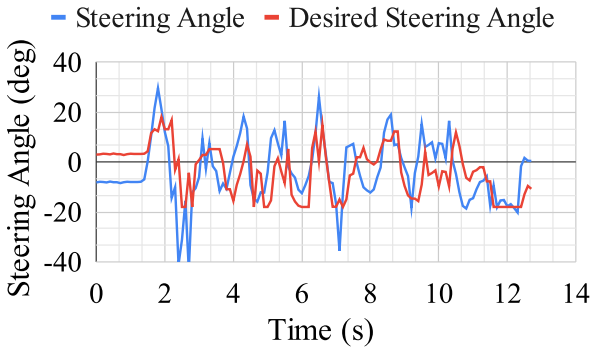

Figure 24 AIRL inference on straight physical track.

One factor that ensured similar results in simulation and physical environments was not attempting to model sensors using RL. This is a common failure point for end-to-end RL models [27]; using a separate object detection module has potential for more robust RL implementation in the physical world.

6. Conclusions

F:SAE-47 need to begin the development of a driverless vehicle to compete in the impending F:SAE driverless class. The following conclusions are drawn from testing SAC and AIRL RL algorithms for this purpose:

- SAC converges faster than AIRL when training in simulation (735 vs 1386 episodes).
- The fastest and most physically-representative training results are achieved with steering angle limits of $\pm 18°$, a steering rate limit of $\pm 112.5°/s$ and using ± 1 as cone colour identifiers. AIRL is trained with a rollout length of 1,000 steps.
- AIRL-trained models perform more reliably than SAC-trained models in simulated inference with 100% median completion on both the training track and a different track. SAC achieves 100% median completion on the training track and 70% on an unknown track.
- AIRL-trained models can perform at a higher speed of 6 m/s than SAC-trained models (capped at 5 m/s) when both are trained at the same speed (4 m/s). Similarly, at the same speed of 4 m/s, AIRL-trained models can function at a longer time-step of 0.55 s vs SAC's 0.3 s.
- An AIRL-trained model performs slightly more reliably on scaled-down simulation tracks (100% track completion rate vs 92.5% from SAC).
- An AIRL-trained model performs slightly more reliably when transferred to the physical world with a 95% full track completion rate compared to 87.5% from SAC.

With a mean steering angle derivative demanded from SAC of $57.6°/s^2$ and AIRL of $56.9°/s^2$, neither algorithm is ready for transfer to a real F:SAE vehicle (smooth expert is $4.4°/s^2$). It is recommended that the F:SAE team first implement classical local planning and control methods (such a pure pursuit) as a baseline, while in parallel continuing to develop both RL approaches. AIRL shows promise of good performance in the shorter term, while SAC has the potential to outperform both AIRL and traditional methods in the longer term.

7. Suggestions for Future Work

Potential exists to improve the performance of the SAC and AIRL RL algorithms. The main areas for further investigation are:

- Test training performance with varied back-end ANN parameters such as learning rate, number of hidden layers and number of neurons per hidden layer.
- Test the performance of the of the reward Functions 2 and 3.
- Attempt to train AIRL on a hard-coded racing line rather than the centre line. Also attempt to add more exploration in the expert's actions for more robust performance.
- Trial common training techniques to improve robustness such as transfer learning (train with multiple tracks of different features, scales and at different speeds) and reward shaping.
- Allow varied vehicle speed during a lap both through relating speed to the selected steering angle and by allowing the RL algorithms to select it directly.
- Test algorithms on a full-scale drive-by-wire F:SAE vehicle with real competition cones marking the track.

References

[1] Formula Student Germany, "FSG: Rules Important Documents." [Online]. Available: https://www.formulastudent.de/fsg/rules/

[2] ——, "FSG Competition Handbook 2021," Formula Student Germany, Tech. Rep., 2021. [Online]. Available: https://www.formulastudent.de/time.

[3] K. Iagnemma and M. Buehler, "Editorial for Journal of Field Robotics - Special Issue on the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 655–656, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20154

[4] ——, "Editorial for Journal of Field RoboticsâSpecial Issue on the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 8, pp. 461–462, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20142

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, sep 2006. [Online]. Available: www.interscience.wiley.com

[6] J. V. Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation research. Part C, Emerging technologies*, vol. 89, pp. 384–406, apr 2018. [Online]. Available: http://dx.doi.org/10.1016/j.trc.2018.02.012

[7] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines*, vol. 5, no. 1, 2017. [Online]. Available: https://doi.org/10.3390/machines5010006

[8] "VRUNet: Multi-Task Learning Model for Intent Prediction of Vulnerable Road Users - Scientific Figure on ResearchGate." [Online]. Available: https://www.researchgate.net/figure/Modules-in-an-autonomous-driving-pipeline_fig2_342886769

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, may 1996. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10166

[10] Spinning Up, "Kinds of RL Algorithms," 2018. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#a-taxonomy-of-rl-algorithms

[11] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuška, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1291–1307, 2012.

[12] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition (2019)*, 2nd ed. O'Reilly, 2011, vol. 44, no. 8.

[13] S. J. and others Ng, Andrew Y and Russell, "Algorithms for inverse reinforcement learning," 2000, p. 2.

[14] B. Piot, M. Geist, and O. Pietquin, "Bridging the Gap Between Imitation Learning and Inverse Reinforcement Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1814–1826, 2017.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," dec 2013. [Online]. Available: https://arxiv.org/abs/1312.5602v1

[16] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. Perez, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[17] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, oct 2021.

[18] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 737–744, dec 2020.

[19] T. H., J. NI, and J. HU, "Autonomous Driving System Design for Formula Student Driverless Racecar," in - *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1–6.

[20] Thomas Bräunl, "FORMULA SAE," 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-38897-3_14

[21] T. H. Drage, J. Kalinowski, and T. Bräunl, "Development of an Autonomous Formula SAE Car with Laser Scanner and GPS," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2652–2657, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016420100

[22] M. Zeilinger, R. Hauk, M. Bader, and A. Hofmann, "Design of an Autonomous Race Car for the Formula Student Driverless (FSD)," 2017.

[23] M. d. l. I. Valls, H. F. C. Hendrikx, V. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. R. Gawel, M. Bürki, and R. Siegwart, "Design of an Autonomous Racecar: Perception, State Estimation and System Integration," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2048–2055, apr 2018. [Online]. Available: http://arxiv.org/abs/1804.03252

[24] N. Gosala, A. Bühler, M. Prajapat, C. Ehmke, M. Gupta, R. Sivanesan, A. Gawel, M. Pfeiffer, M. Bürki, I. Sa, R. Dubé, and R. Siegwart, "Redundant Perception and State Estimation for Reliable Autonomous Racing," in - *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6561–6567.

[25] A. Dhall, D. Dai, and L. Van Gool, "Real-time 3D traffic cone detection for autonomous driving," in *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2019-June. Institute of Electrical and Electronics Engineers Inc., jun 2019, pp. 494–501.

[26] J. Kabzan, M. d. l. I. Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari,

N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, "AMZ Driverless: The Full Autonomous Racing System," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, may 2019. [Online]. Available: https://arxiv.org/abs/1905.05150v1

[27] D. Zadok, T. Hirshberg, A. Biran, K. Radinsky, and A. Kapoor, "Explorations and Lessons Learned in Building an Autonomous Formula SAE Car from Simulations," *CoRR*, vol. abs/1905.0, 2019. [Online]. Available: http://arxiv.org/abs/1905.05940

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *35th International Conference on Machine Learning, ICML 2018*, vol. 5, pp. 2976–2989, jan 2018. [Online]. Available: https://arxiv.org/abs/1801.01290v2

[29] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, oct 2017. [Online]. Available: http://arxiv.org/abs/1710.11248

[30] A. Ezquerro, M. Angel Rodriguez, and R. Tellez, "openai ros," jun 2021. [Online]. Available: http://wiki.ros.org/openai_ros

Appendix A FSG Points Breakdown

CV = combustion, EV = electric, DV = driverless vehicle, DC = driverless cup

	CV & EV	DC
Static Events:		
Business Plan Presentation	75 points	-
Cost and Manufacturing	100 points	-
Engineering Design	150 points	150 points
Dynamic Events:		
Skid Pad	50 points	-
DV Skid Pad	75 points	75 points
Acceleration	50 points	-
DV Acceleration	75 points	75 points
Autocross	100 points	-
DV Autocross	-	100 points
Endurance	250 points	-
Efficiency	75 points	-
Trackdrive	-	200 points
Overall	1000 points	600 points

Table A4 FSG points allocation by event [1].

Appendix B FSG Driverless Tracks

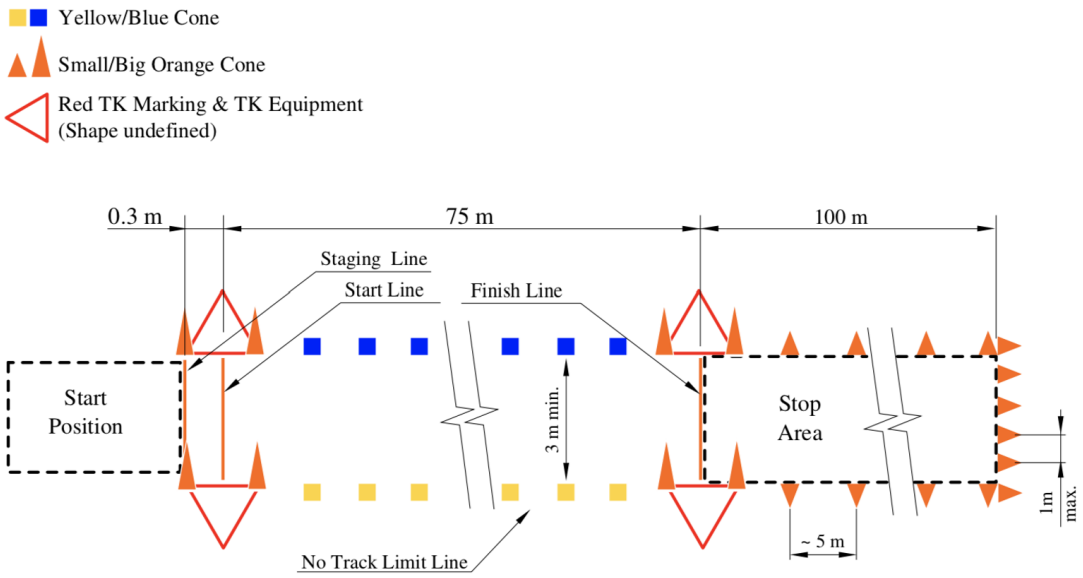

Figure B25 FSG acceleration track specifications [2].

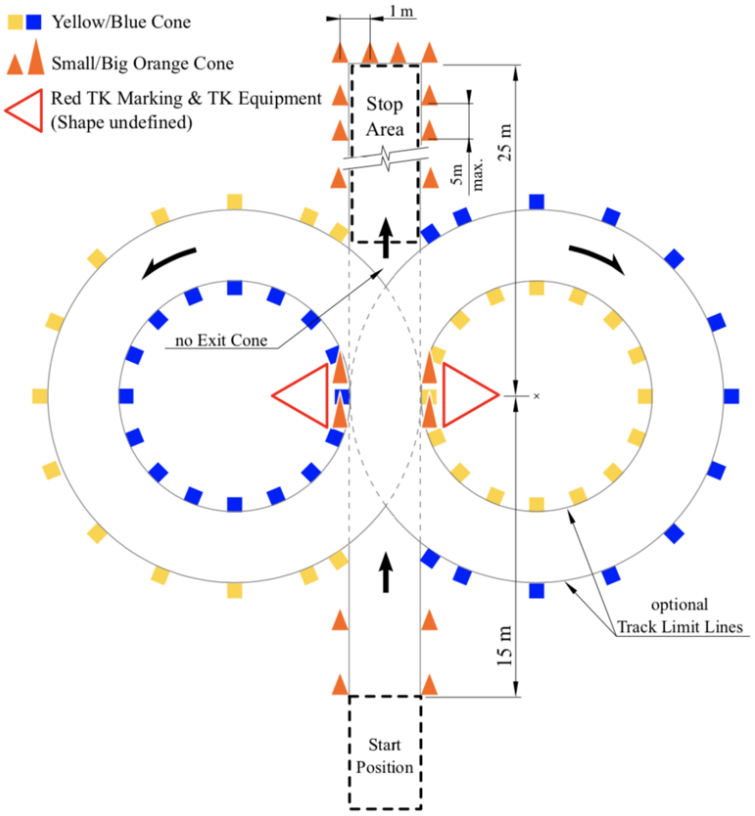

Figure B26 FSG skidpan track specifications [2].

Appendix C Parameter List for ANN Training in RL Algorithms

SAC

- **Actor and critic learning rate -** 3e-4
- **Actor and critic learning rate α -** 3e-4
- **Actor and critic hidden layer nodes -** (256, 256)

AIRL

- **Actor, critic and discriminator learning rate -** 3e-4
- **Actor and critic hidden layer nodes -** (64, 64)
- **Discriminator hidden layer nodes -** (100, 100)

Appendix D Complete List of Code Repositories Utilised

- **Custom combined repository** - https://github.com/HarveyMerton/uoa_fsd
- **Formula student simulator** - https://github.com/AMZ-Driverless/fssim
- **Formula student driverless skeleton** - https://github.com/AMZ-Driverless/fsd_skeleton
- **Reinforcement learning algorithm implementations** - https://github.com/ku2482/gail-airl-ppo.pytorch
- **CARES custom ROS messages** - https://github.com/UoA-CARES/cares_msgs
- **Pylon stereo camera image publishing** - https://github.com/UoA-CARES/pylon_camera
- **Stereo camera calibration** - https://github.com/UoA-CARES/stereo_calibration
- **Stereo camera marker detection** - https://github.com/maraatech/aruco_detector

Appendix E Code Modules Overview

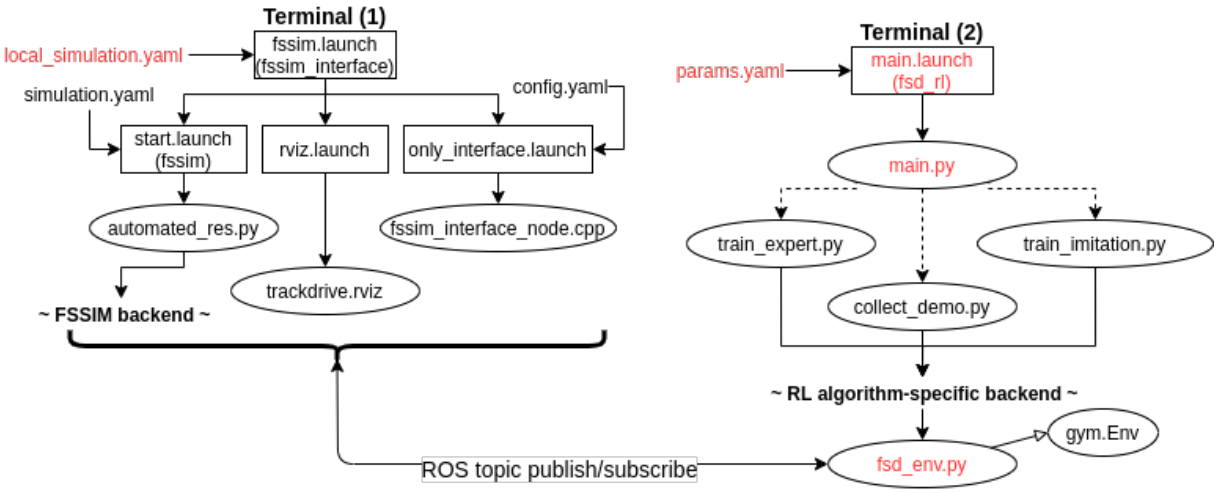

Figure E27 Dependency graph - code.

Key: Rectangles represent launch files, ovals are classes, parameter files have no boarder while labels in brackets are ROS package names where the files do not belong to the same package as the parent. Labels in red are the main files altered in this project.

Appendix F Comparison of FS Simulator and Physical Setup Parameters

Note that the full size simulated vehicle is designed to exactly model AMZ's real F:SAE vehicle (Gotthard driverless). The 'sim scaled' system represents this system scaled down to roughly match the parameters of the physical testing system. Kinematic similarity between the simulated and simulated scaled systems is ensured by halving geometric parameters and speeds where possible.

Parameter	Simulated	Sim scaled	Physical
Vehicle			
Camera			
Position X (m)	N/A	N/A	0.15
Position Y (m)	N/A	N/A	0.08
Position Z (m)	N/A	N/A	1.00
Depression angle around Y (deg)	N/A	N/A	35
Vehicle geometry			
Track width (m)	1.20	0.60	0.50
Wheel-base (m)	2.44	1.22	0.70
Front axle to turning centre (m)	1.22	0.61	0.70
Seat height above ground (m)	0.26	0.26	0.30
Wheel height - ground to centre (m)	0.40	0.4	0.125
Vehicle kinematics			
Turning radius (m)	2.50	1.25	1.50
Steering angle limits (deg)	45 (limited to 18)	45 (limited to 18)	15
Steering angle backlash (deg)	0	approx. 0	approx. 0
Steering ratio	N/A	N/A	1:1
Track			
Cones			
Height: base to top (m)	0.33	0.33	0.34
Height: base to marker centre (m)	N/A	N/A	0.19
Cone placement			
L/R (m)	3.30	2.00	1.25
L/L (m)	2-4	1.00	1.00
R/R (m)	2-4	1.00	1.00
Sensors			
Cone position noise - r (μ, σ)	$0, 0.2$	$0, 0.1$	0, approx. 0
Cone position noise - $\theta\,(\mu, \sigma)$	$0, 0.007$	$0, 0.0035$	0, approx. 0
Steering angle sensor accuracy (%)	100	100	94
Camera range: max (m)	10	5	4
Camera range: min (m)	0.5	0.25	0.25
Camera field of view (FOV) (deg)	180	180	80

Table F5 Simulator-physical parameter comparison.

Appendix G Videos of Results

- **Just results:** https://www.youtube.com/watch?v=bpdDmsVrhok
- **Background and results with voice:** https://www.youtube.com/watch?v=0PPg5hFWtp8

Appendix H Scaled Track Images

Tracks start from pose shown and end at pink finish line.

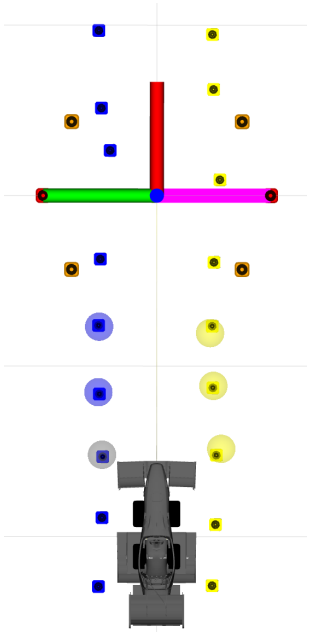

Figure H28 Simulated track: straight.

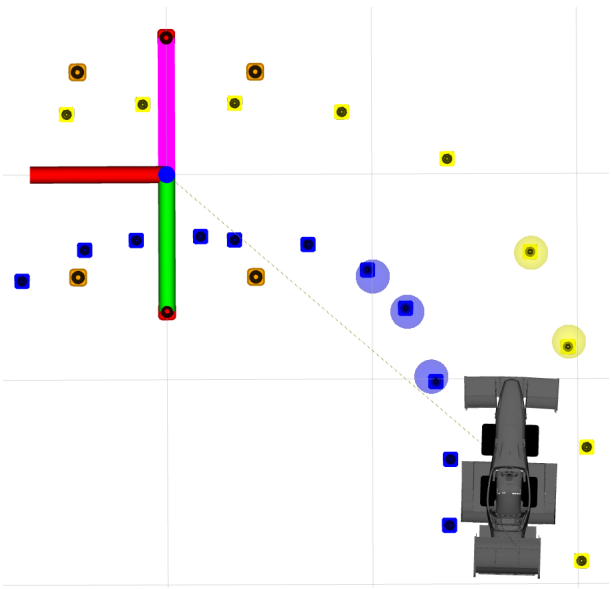

Figure H29 Simulated track: left.

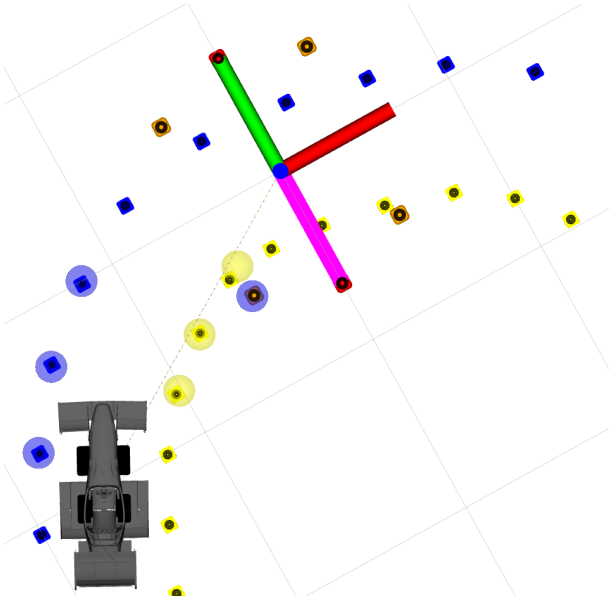

Figure H30 Simulated track: loose right.

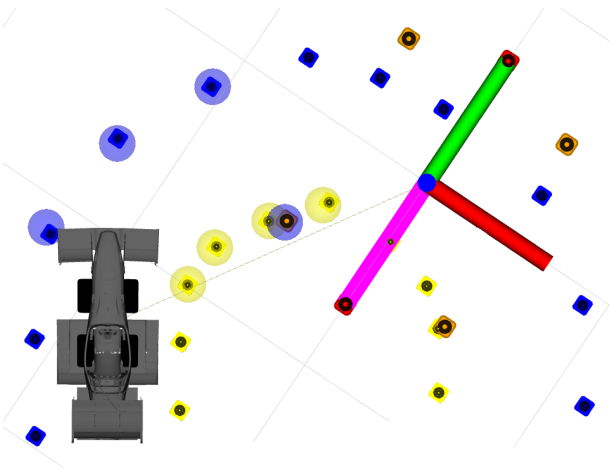

Figure H31 Simulated track: tight right.

Appendix I Physical Setup Images

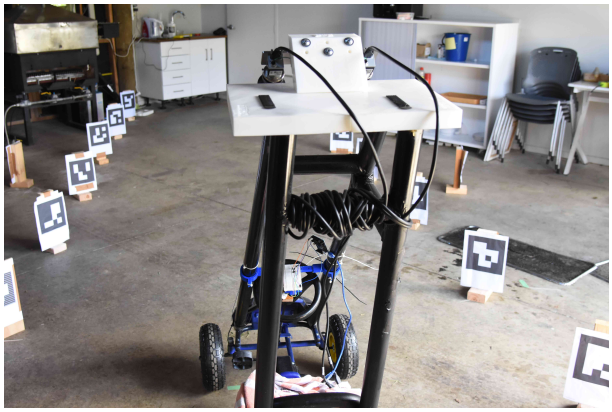

Figure I32 Physical test kart on straight track.

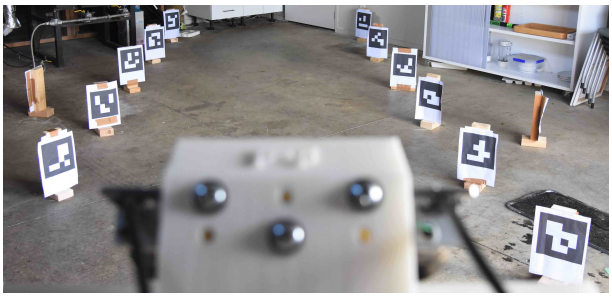

Figure I33 Physical straight track - platform view.

Figure I34 Physical stereo camera.

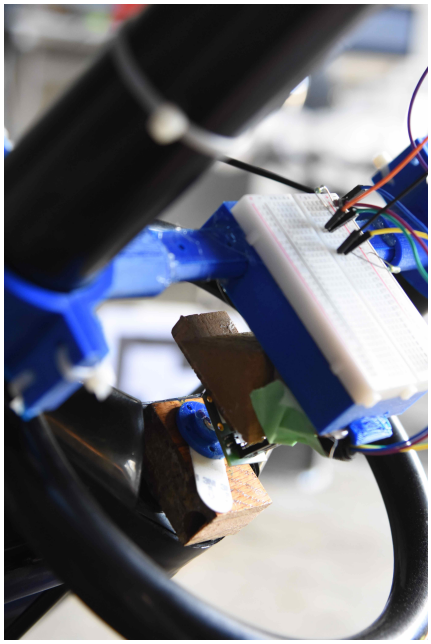

Figure I35 Physical steering angle sensor.

Appendix J Stereo Camera Cone Marker Detection

Figure J36 ArUco marker detection through stereo cameras.

Appendix K Equipment Description

- **Primary PC:** Dell Latitude E7250 with Intel Core i5-5300U (dual core, 2.3 GHz) and 8 GB RAM.
- **Stereo camera:** two Balser acA1920-40uc cameras with a Sony IMX249 color CMOS sensor and a global shutter. Capable of capturing 41 frames per second at 2.3 MP resolution.
- **Steering angle sensor:** AS5600-SO_EK_AB magnetic rotary position sensor.

Appendix L Steering Angle Sensor Setup

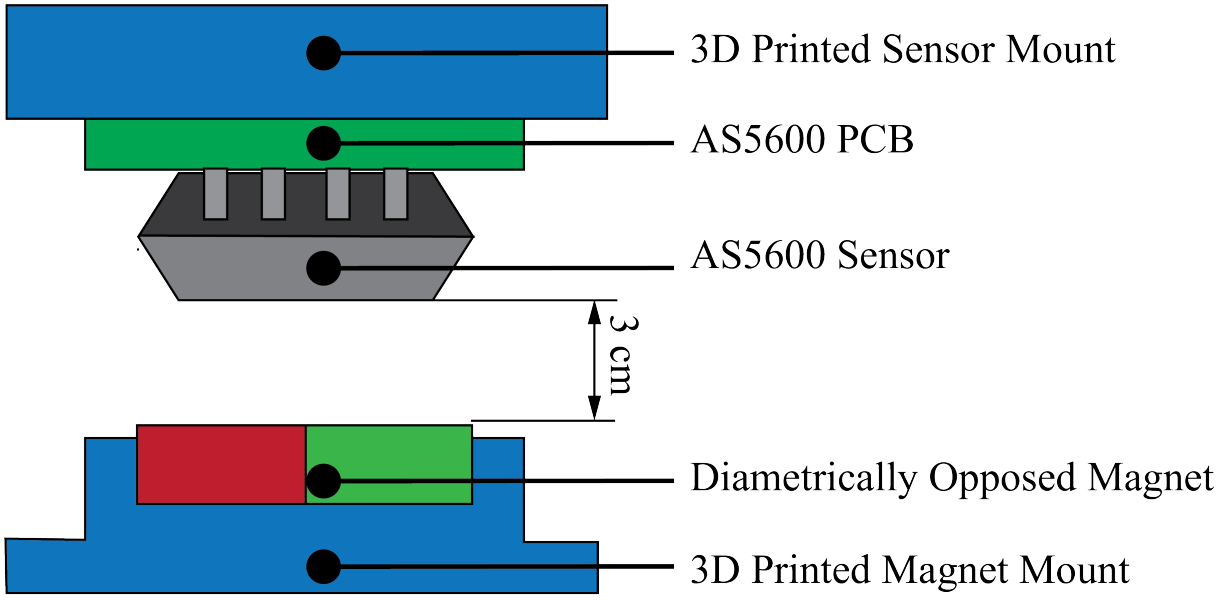

Figure L37 Steering angle sensor setup.