BRIDGING HUMAN EMOTION AND AUTONOMOUS VEHICLE CONTROL: REINFORCEMENT LEARNING ENHANCEMENTS IN THE CARLA SIMULATOR

A Thesis

by

TIMOTHY MARION JUDE LYONS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Mechanical Engineering

The University of Texas Rio Grande Valley

August 2024

BRIDGING HUMAN EMOTION AND AUTONOMOUS VEHICLE CONTROL:

REINFORCEMENT LEARNING ENHANCEMENTS

IN THE CARLA SIMULATOR

A Thesis
by
TIMOTHY MARION JUDE LYONS

COMMITTEE MEMBERS

Dr. Mohamadhossein Noruzoliaee
Co-Chair of Committee

Dr. Constantine Tarawneh
Co-Chair of Committee

Dr. Horacio Vasquez
Committee Member

Dr. Fatemeh Nazari
Committee Member

August 2024

ABSTRACT

Lyons, Timothy M.J., Bridging Human Emotion and Autonomous Vehicle Control: Reinforcement Learning Enhancements in the CARLA Simulator. Master of Science in Engineering (MSE), August 2024, 52 pp., 3 tables, 23 figures, 23 references.

Autonomous vehicles are a key component for an evolution in transportation as they may lead to increased safety and productivity. As autonomous vehicles advance, industries will look to further increase customer satisfaction through personalized experiences for passengers. Reinforcement learning may be the key to success for autonomous vehicles. The development of autonomous vehicles utilizing reinforcement learning, plays a key role in the advancements of transportation.

This thesis presents a unique approach to autonomous vehicles developed with reinforcement learning. A conceptual framework to include human facial emotion for autonomous vehicles developed with reinforcement learning is proposed. Surrounding vehicles are added to the testing environment. Dynamic speed limits are added to the environment. Brakes are added to the vehicle controls as an additional action for the agent.

DEDICATION

This thesis would not have been possible without the love and support of my family. My loving wife, Caitlin Lyons, your love and support is what helped me get this far. My father, Patrick Lyons, and my mother, Mary Lyons, thank you for never giving up on me and giving me this opportunity. My brother Paddy, thank you for being a role model and motivating me to be who I am today. To my fellow classmate Joshua Ontiveros, to all the late nights studying, cheers bud.

ACKNOWLEDGEMENTS

I appreciate the opportunity given to me through the National Science Foundation CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS), funded by NSF Award No. 211650.

I want to thank Dr. Mohamadhossien Noruzoliaee who provided me with this opportunity to obtain a Master's degree and achieve interdisciplinary knowledge in a state of art field of research. Your guidance though my journey, patience with my research and mentorship has been a key role in my success.

I am thankful to Dr. Constantine Tarawneh for supporting me through my journey. Thank you for pushing me to further my education and providing me with path to follow.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Artificial intelligence is a highly competitive field of study. A wide variety of methods and algorithms have been developed and implemented into a variety of applications. Advancements in artificial intelligence has led to the advancements in autonomous vehicles. Autonomous vehicles may lead to safer driving environments, fewer accidents and an increase in productivity for those utilizing the benefits of autonomous vehicles. A key role in autonomous vehicle advancements is the implementation of reinforcement learning.

Problem Statement

Autonomous vehicles continue to advance and with the advancements, come opportunities for new technological implementations to manipulate how autonomous vehicles will operate. The goal of this research is to develop a conceptual framework to bridge the gap between an autonomous vehicle and the human emotion of the passenger. There are driving situations in which some human passengers are comfortable, while other human passengers may not feel comfortable. Theoretically it would be possible for an autonomous vehicle to consider these human emotions, and correct its driving behavior to make the passenger more comfortable. A simulated environment will need to be developed in order to safely test this theory. An existing repository will be modified to create driving scenarios in an attempt to create more realistic scenarios that may draw out more human emotions. These emotions will ultimately contribute to the conceptual framework suggested to alter the behavior of a vehicle, which has been developed using reinforcement learning.

Motivation and Contribution

The autonomous vehicle industry is constantly growing. One day fully autonomous vehicles may be on the road, whether they be personal vehicles or taxis utilized by companies. A key advancement and selling point for autonomous vehicles is that they may be a personalized experience for the passenger in a sense that the vehicle will adjust its driving behavior based on the emotion of the passenger. As an engineer, the number one factor is safety. The implementation of a vehicle which prioritizes safety while also considering the emotions of the passenger would be a state-of-the-art advancement in the autonomous vehicle industry. Contributions are as follows:

- A conceptual framework to include human facial emotion for autonomous vehicles developed with reinforcement learning.
- Surrounding vehicles are added to the simulation in order to create more dynamic scenarios, bringing the simulation closer to real-world scenarios.
- Dynamic speed limits are added to the environment to create more driving scenarios which may extrude reactions from a simulated passenger.
- Brakes are added to an autonomous vehicle which only utilizes throttle and steering in order to give the vehicle more realistic controls.

Thesis Organization

The following thesis will be organized in the following way. Chapter II will contain a literature review. This section will cover an overview of reinforcement learning, autonomous vehicle control, reinforcement learning applications in autonomous vehicles and the details important to said application. These details will include states, actions and rewards used for autonomous vehicle applications, which include reinforcement learning, as well as the simulators

and scenarios created for these applications. Chapter III will include the methodology for this thesis. The reinforcement learning algorithm being utilized, Proximal Policy Optimization, will be covered. The actor-critic architecture will be discussed. A conceptual framework to include human emotion be proposed. Theoretical consideration to implement said conceptual framework will also be included. Chapter IV will cover the model implementation and results. CARLA simulator will be covered as well as the modifications made to the simulation. The models developed, trained and tested will be compared and the results from the testing will be reviewed. Chapter V will include the conclusion and any future work that may be recommended.

CHAPTER II

LITERATURE REVIEW

Reinforcement Learning

Reinforcement learning is a branch of artificial intelligence and is considered to be a form of unsupervised learning. A general description given to reinforcement learning is that reinforcement learning correlates situations to actions, and learns optimal actions in these situations to develop an optimal policy (Sutton et al., 2018, p.1). The term agent refers to the learner in reinforcement learning, which may include a robot, video game character, or an autonomous vehicle. Agents are developed with a goal orientated methodology. The agent must learn, through experience, how to optimize its performance to achieve a goal. Agents perform in what is known as an environment. The environment considers the world in which the agent is in, such as surrounding structures, obstacles, and weather. Agents start in a state, which is the current position of the agent and the current position of the surrounding environment. The agent will take actions based on the state of the environment, and once the agent takes an action based on the state of the environment, the agent will move into a new state. After the agent takes an action, the agent will receive a reward based on the new state in which the agent is in. The goal of the agent is to take the best actions, depending on the its state in the environment, to maximize

rewards and develop an optimal policy. A policy is the mapping between states and actions which the agent will follow to achieve the goal set for the agent (Sutton et al., 2018, p.6).

Acknowledging that agents learn through experience, it is important to note the trade-off between exploration and exploitation. As an agent learns through rewards, the agent may begin to exploit this knowledge. The resultant of the behavior can be detrimental to the learning process if the agent does not balance exploitation and exploration. Since agents learn through experience, they also need to explore different actions in similar states to learn the optimal actions to take. Furthermore, an agent cannot become hyper focused on the current reward, but instead, may also need to consider future rewards which may be affected by the current action the agent is taking. The reward function is used to give immediate rewards for the agent. The advantage function is introduced to assist in the agent considering future reward. While rewards fall into a primary consideration, with the implementation of the advantage function, future rewards are also considered by the agent (Sutton et al., 2018, p.6).

The agent must ultimately learn how to model the environment it is in. The methods of modeling environments can be broken down into two categories, known as model-free and model-based (Sutton et al., 2018, p.7). If an environment is simple enough to be modeled, a model-based method may be used. Utilizing model-based methods allow for planning, as in the decision making of the agent is predetermined. In a more complex environment, model-free methods must be used. Model-free methods are trial and error driven, as the agent must experience as many situations as possible to develop an optimal policy (Sutton et al., 2018, p.7). A good analogy for these methods can be taken from developing an agent to play checkers or chess. The game of checkers can easily be modeled due to the limitations of moves an agent can take, thus reducing the states in which an agent can be in. Chess on the other hand has a much

greater number of states that an agent may be in due to the diversity of options in the moves one can make. Due to the limitations in checkers one can easily use a model-based strategy, whereas in chess a model-free strategy would be the optimal choice.

Markov Decision Processes (MDPs) assist in providing a simplified structure for learning through interactions. The purpose of learning through interactions is to achieve a specific goal. The "agent" refers to the learner while the "environment" refers to everything external to the agent. Continuous interactions occur between the agent and the environment, where the agent chooses actions and the environment responds by providing new situations and rewards. The agent then aims to maximize the rewards which are numerical values provided to the agent that guide the agent through its learning process (Sutton et al., 2018, p.47).

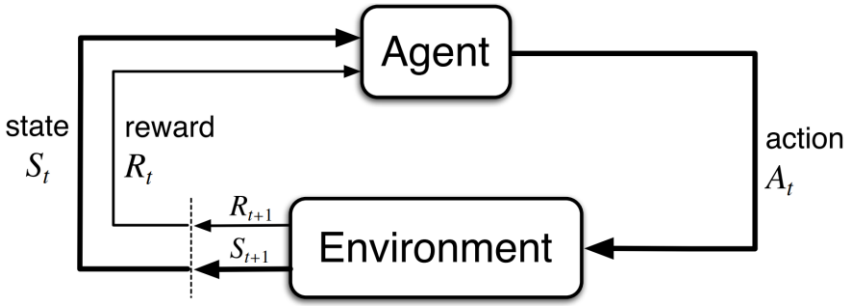

Figure 1: Agent Environment in an MDP

According to (Sutton et al., 2018, p.48), agents developed via the Markov Decision Process interact with the environment through discrete timesteps, t. The agent observes the state, S_t, of the environment for every timestep (t = 0, 1, 2, 3, …). Based on the current state, the action will then select an action, A_t, where the action comes from a set of possible action A(s). The resultant is a reward, R_{t+1}, provided by a reward set R. The agent then transitions into a new state, S_{t+1}. This process of interaction continues and develops a pattern of:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

(Sutton et al., 2018, p.48), explains that finite MDPs are limited to a finite number of elements when it comes to the state, action and reward sets. Consequently, the random variables which are state and reward, have a well-defined discrete probability distribution. This distribution depends solely on the previous state and action. Therefore, the probability of transitioning to a state S_0 in S, and receiving a reward r in R at time t, can be expressed through the probability distribution:

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

This formula defines the probabilistic dynamics of the system, where the likelihood of moving to a specific state and receiving a certain reward is conditioned on past actions and states (Sutton et al., 2018, p.48).

The Bellman equation assists in determining an optimal policy by simplifying decision-making problems which would otherwise be too complex. A number of fundamental reinforcement learning algorithms such as Q-Learning and Value Iteration were developed thanks to the path paved by the Bellman equation. A key component in reinforcement learning algorithms is the value function. An important role of the Bellman equation, is its assistance in creating the value function. The value function was developed as a result of the Bellman equation and can be used to determine how beneficial it is for an agent to be in a given state (Sutton et al., 2018, p.60). The value function equation is given as:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v_\pi(s')]$$

Where $v_\pi(s)$ is the value function in a policy π, $\pi(a|s)$ is the probability that an action a is chosen in a state s under the current policy. The probability of moving to the next state s' and receiving a reward r, after the action is taken in the current state is given by $p(s', r \mid s, a)$. The discount factor, γ, is used to value future reward with respect to immediate reward (Sutton et al., 2018, p.73-74).

Expected rewards are also used to assist in calculating the value function and determining the best fit policy for a reinforcement learning model. Calculating future rewards assists in creating a balance between immediate rewards and future rewards. The reward function, given by (Sutton et al., 2018, p.49), to compute expected rewards for state-actions pairs is as follows:

$$r(s,a) = \sum_{r \epsilon R} r \sum_{s' \epsilon S} p(s', r|s, a)$$

The forementioned formulas and architecture that make up a baseline explanation MDPs still require some form of modeling of the environment. The introduction of Q-learning, enables more complex environments to be handled. This breakthrough is provided by (Sutton et al., 2018, p.131) and is defines as:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

This implementation contributes to the simplification of the analysis needed for the algorithm, leading to early convergence. The contribution of Q-learning in combinations with special cases of TD-methods, introduced the development of model-free methods (Sutton et al., 2018, p.138).

At a surface level reinforcement learning is already complex. Developing a goal orientated agent takes many considerations. The environment of the agent, the actions the agent is able to take, and the reward structure for the agent must all be methodically determined. There are a number of reinforcements learning algorithms, each with their own unique properties. The application, strategy, and implementation must all be considered when determining which reinforcement learning algorithm will be the best fit.

Autonomous Vehicle Control

Autonomous vehicles have found their way to popularity in research and develop over the past few years. It is estimated that 90% of car accidents are due to human error rather than vehicle failure, which only contribute to roughly 2% of accidents involving a vehicle (Kuutti et

al., 2019, p.1). Contributions of autonomous vehicles also include better fuel consumption, reduced pollution via car sharing, an increase in productivity and improvements to the flow of traffic (Kuutti et al., 2019, p.1). Methods which involve modeling driving behavior has proven to be a difficult task, as the gender, age, driving experience, mood and other contributing factors from the driver, all have an effect on the driving behavior (You et al., 2019, p.1). Therefore, by eliminating the involvement of a human driver, and developing fully autonomous vehicles has led to the rise in autonomous vehicle research. Control techniques, optimization methods, machine learning and reinforcement learning are a few fields of studies which are being explored for the development of autonomous vehicles (Aradi, 2020, p.1). The three contributing factors to autonomous vehicle control include perception, planning and control. The typical architecture for autonomous vehicle control, provided by (You et al., 2019, p.2) can be seen below:

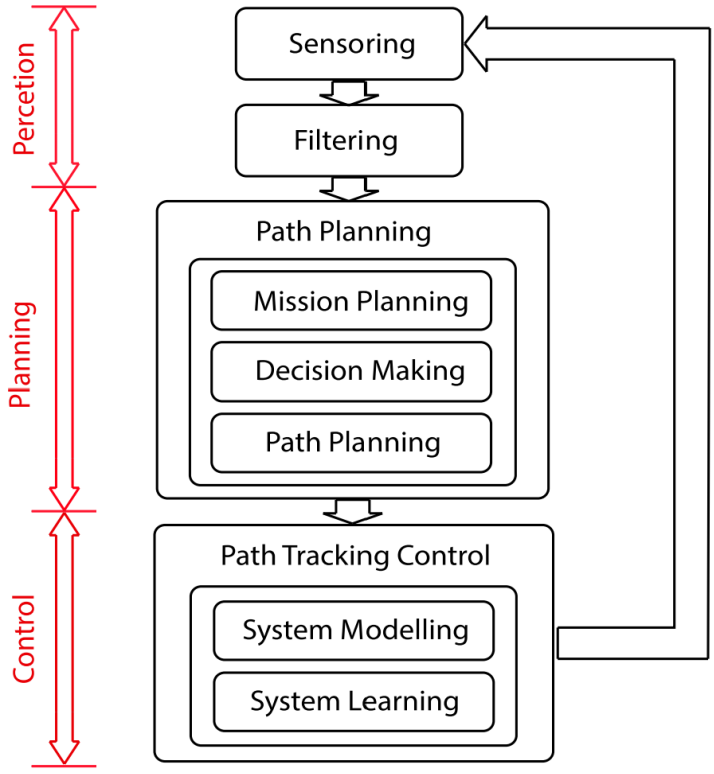

Figure 2: Autonomous vehicle control architecture

Beginning with perception, this methodology gives the autonomous vehicle vision of its surrounding environment. The use of sensors such as RGB cameras and LiDAR assist as sensors to determine the state of the vehicle in its current environment. Path planning consists of a goal orientated task, where an objective, such as a specific traffic scenario is given to the autonomous vehicle. Decision making consists of actions the vehicle is able to make, such as steering, throttle and braking. Path planning involves giving the vehicle a designated route to follow, whether it is in an urban city or highway scenario, this method assists in the direction the vehicle is meant to follow. Path tracking control includes the monitoring of the path planning level by obtaining signals and maintaining the vehicles stability while tracking the vehicles desired path.

There are three basic sensors that are used to perceive the environment around the vehicle. These three sensors can be broken down to cameras, LiDAR and radars (Ignatious et al., 2022, p.738). The two most commonly used sensors are cameras and LiDAR. Cameras are most commonly used as they provide clear images, are generally inexpensive compared to other technologies, and can reveal the identity of both static and dynamic objects. LiDAR is used to detect reflections from infrared or laser pulses produced by the LiDAR which reflect off of other objects. This enables the LiDAR to produce cloud point data in either a 1D, 2D or 3D environments, which provides information on the surrounding area. There are challenges associated with sensors, including reliability and accuracy. Sensors are also limited in their abilities, which is why in most cases, sensor fusion is used to optimize perception capabilities (Ignatious et al., 2022, p.738). A table provided by (Ignatious et al., 2022, p.740), including camera types and their performance capabilities can be seen below:

Table 1: Sensor Comparison

Factors	Camera	LiDAR	RADAR	Fusion
Range	--	--	✓	✓
Resolution	✓	--	×	
Distance Accuracy	--	✓	✓	✓
Velocity	--	×	✓	✓
Color Perception, e.g Traffic lights	✓	×	×	✓
Object Detection	×	✓	✓	✓
Object Classification	✓	×	×	✓
Lane Detection	✓	×	×	✓
Obstacle Edge Detection	✓	✓	×	✓
Illuminations Conditions	×	✓	✓	✓
Weather Conditions	×	--	✓	✓

Path planning and path tracking have proven to be a difficult task due to the decision making and control aspects involved in the process. This is due to the vehicle being nonlinear and non-holonomic (Liu et al., 2017, p.1). Maintaining the desired speed, keeping passengers comfortable, while at the same time avoiding collisions is a task that is hard to maintain. Methods such as graphs-based approaches, sampling-based methods and trajectory optimization-based approaches. such as Model Predictive Control have all been used in an attempt at uncovering the best methods, or combination of methods, for autonomous vehicle path planning (Liu et al., 2017, p.1). Model Predictive Control has been utilized in multiple works, (Liu et al., 2017) and (Berntorp, 2017), however, even with recent advancements there is still a need for optimization in order to develop a robust path planning technique.

When considering a pipeline for autonomous vehicle control (Kiran et al., 2022) developed a general pipeline for modern autonomous vehicle systems with the following key problems to be addressed, seen below:

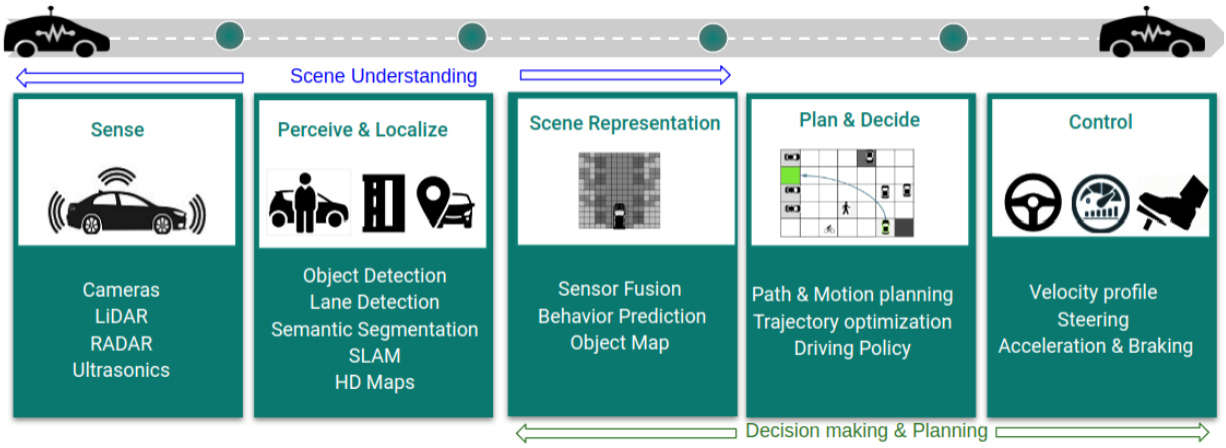

Figure 3: Autonomous driving system pipeline

This figure further elaborates on the modules necessary to complete autonomous vehicle control. As previously mentioned, the understanding of the scene is determined via sensors, object detection methods, a policy on which path and trajectory is to be followed and the controls which will be given to the vehicle. The control of the vehicle can be generalized into three aspects, that is steering, acceleration and brakes.

Reinforcement Learning in Autonomous Vehicle Applications

Control approaches have been implemented based on reinforcement learning methods (Folkers et al., 2019). The framework used in (Folkers et al., 2019, p.1) work, utilizing a control loop, can be seen below:

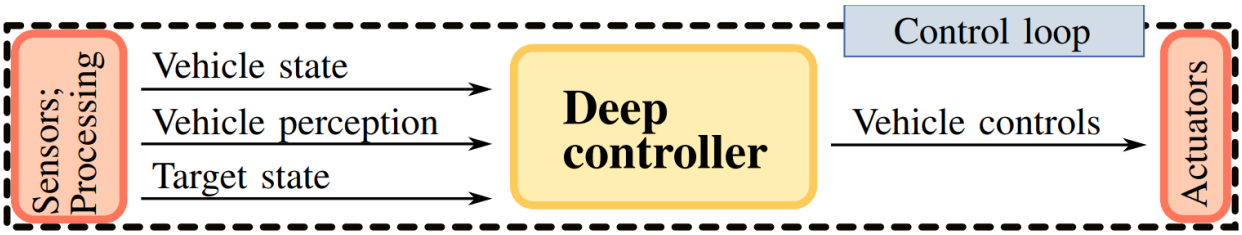

Figure 4: Deep control loop approach

Classic controllers such as PID, MDP and the Fuzzy Control method have proven to have good performance evaluations. However, the conditions of these controllers are typically environment dependent and as a result, it is necessary to constantly fine tune the hyperparameters associated with each method (Pérez-Gil et al., 2022, p.3356). Imitation learning has also been used, and has proven to achieve satisfactory results in a short period of time. The drawback of imitation learning is the limitation of learning from imitating human behavior. Due to the limitations, imitation learning can result in real driving situations that are dangerous, due to some scenarios being non reproduceable by the trainer, such as life and death driving scenarios. An alternative method to developing autonomous vehicle is reinforcement learning. Reinforcement learning involves learning from experience, by introducing this methodology, autonomous vehicles can be trained on multiple scenarios, via simulation, to increase the experiences the vehicle will encounter. This methodology may reduce human involvement and pose a solution to handling complex dynamic environments such as the environment of an autonomous vehicle in an urban setting. Reinforcement learning is also adaptable to different environments. By improving its decision-making overtime the necessity for parameter tuning is reduced, as robust reinforcement learning methods learn to handle a variety of environments. The general concept of including reinforcement learning into training an autonomous vehicle is as follows. Consider the agent in the reinforcement learning strategy to be the autonomous vehicle. The agent performs in an environment, consisting of everything surrounding the vehicle such as the road, traffic signals, surrounding vehicles, pedestrians, etc. The state of the agent includes factors, such as the velocity of the vehicle, the vehicles position and weather conditions. Actions are provided to the agent, in this case it may include actions such as steering, throttle and brake. Rewards, whether they are positive or negative rewards, are also provided to the agent.

Positive rewards may include reaching an optimal speed and maintaining that speed, while negative rewards may include encountering a collision with another vehicle or object. Ultimately a policy is to be developed, which assists in mapping states to actions in order to optimize the potential cumulative reward the agent will receive.

A consideration in developing an autonomous vehicle with reinforcement learning is handling the high dimensional state space involved. The method used to handle high dimensional state spaces is including a neural network. A general feedforward neural network provided by (Sutton et al., 2018, p.224) can be seen below:

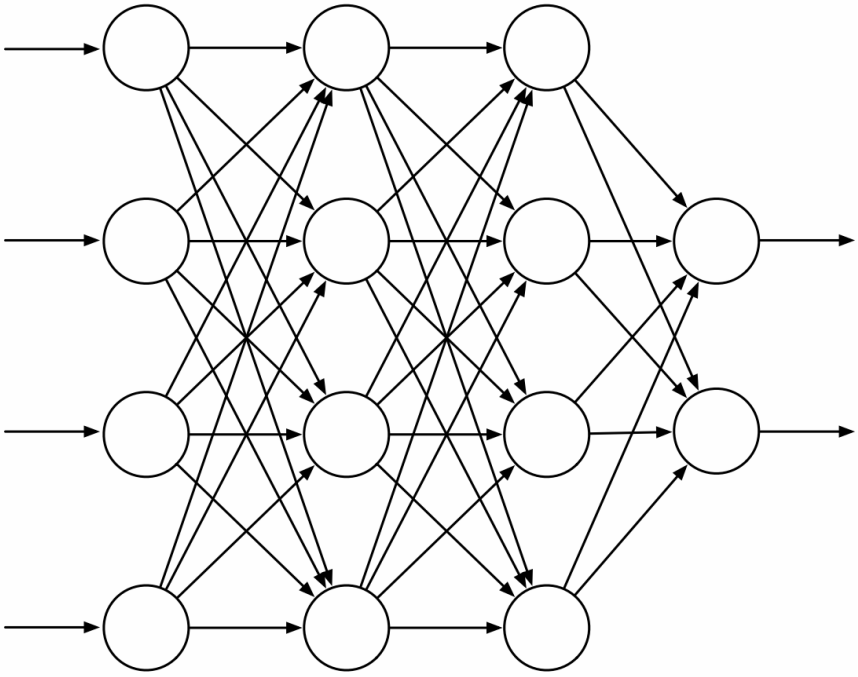

Figure 5: Generic feedforward neural network

The term, deep reinforcement learning includes the involvement of a neural network to handle high dimensionalities. Neural networks are multi-layered networks which model the human brain. Deep reinforcement learning has proven to be a success in multiple training scenarios for autonomous vehicles. Some of these reinforcement learning methods have even led

to real world testing, such as Learning to Drive in a Day (Kendall et al., 2019). A framework for a reinforcement learning application in autonomous vehicles, provided by (Kendall et al., 2019, p.1), can be seen below:

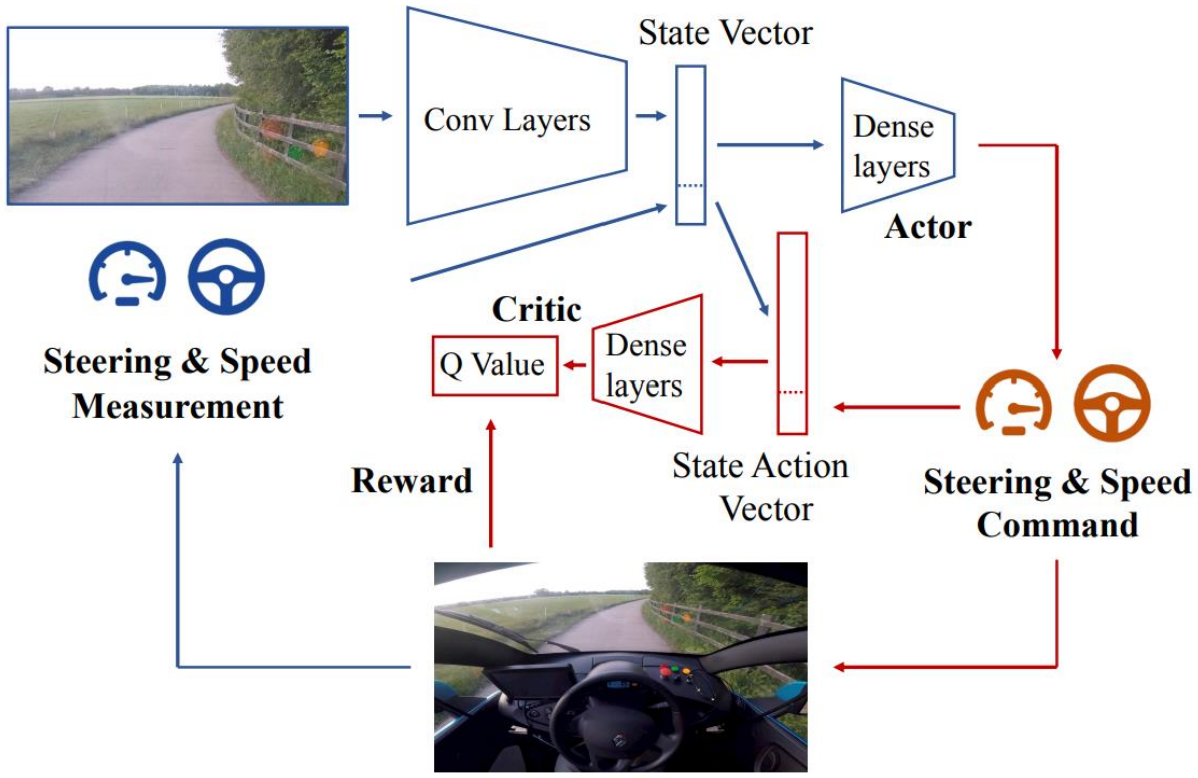

Figure 6: Learning to drive in a day framework

Algorithms such as Deep Q-Networks have been used to train autonomous vehicles (Pérez-Gil et al., 2022). The framework provided by (Pérez-Gil et al., 2022, p.3555) can be seen below:

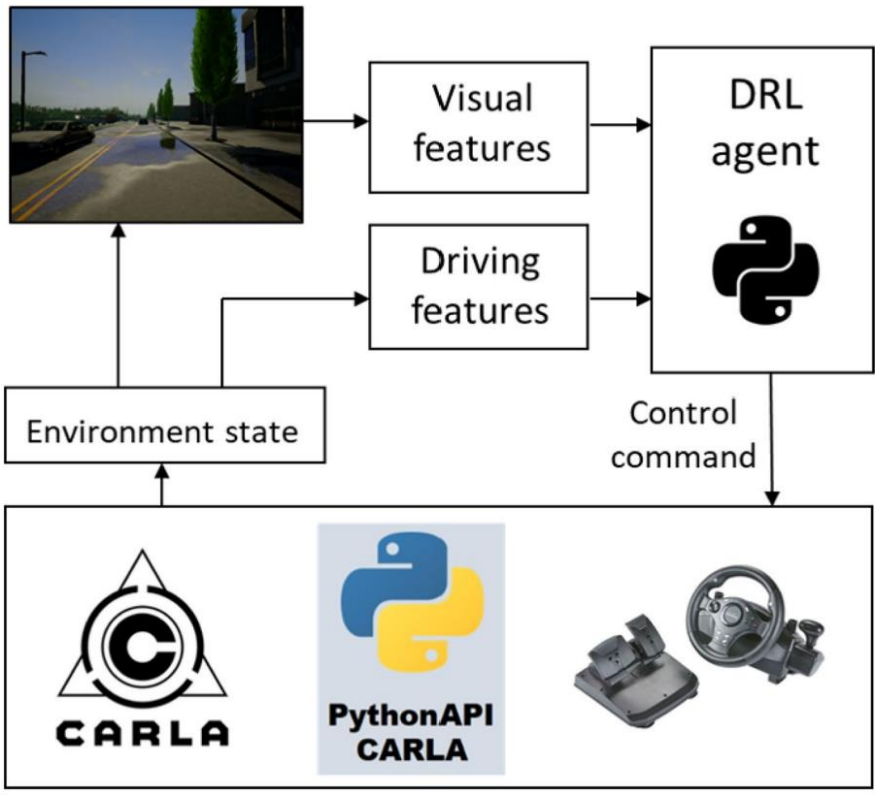

Figure 7: Deep reinforcement learning base control framework

States, Actions and Rewards

The states, action and rewards play a key role in developing an autonomous vehicle when utilizing reinforcement learning. How the states are obtained, via sensors, will determine the ability of the vehicle to perceive the surrounding environment. It is crucial to provide enough information to the vehicle, in a meaningful way, so that the vehicle may learn the appropriate behaviors. The state of the vehicle may include factors such as the vehicles speed, position and orientation, as well as obstacles, traffic signals and weather. This information is typically provided with sensors such as RGB camera, semantic segmentation, LiDAR, radar, GPS, etc. The actions given to the vehicle are typically generalized to three actions including steering, throttle and brake. The steering of the vehicle ranges from [-1,1] where -1 represents a full steering angle to the left, and 1 represent a full steering angle to the right. Throttle and brake

have similar values of [0,1] where 0 represents no throttle or brake and 1 represents full throttle or brake. Some experiments have also considered actions such as gears, including reverse, and a hand brake. If both throttle and brake are being used, it is good practice to generalize throttle and brake into one action being acceleration. Acceleration can then be mapped from [-1, 1] and these values then mapped to throttle and brake where throttle is [0,1] and brake is [-1,0]. Rewards assist in the ability of the agent to learn the correct policy, depending on the goal orientated objective of the application. It is common to include terminating factors in the form of negative rewards such as, collisions, traffic infractions and lane deviation. Furthermore, rewards may include maintaining speed, path following, and reaching a goal. Some examples of autonomous vehicle experiments developed with reinforcement learning can be seen below, including the source, algorithm used, states, actions and rewards.

Table 2: Algorithm, state, action and reward comparison

Source	Algorithm	States	Actions	Rewards
Learning to drive in a day	DDPG	Monocular camera image, vehicle speed, steering angle	Steering angle, Speed	Forward Speed, Traffic Rule Infraction Termination

Table 2 (cont.)

Source	Algorithm	States	Actions	Rewards
Deep reinforcement learning based control for Autonomous Vehicles in CARLA	DQN, DDPG	Visual features, waypoints, vehicle speed, distance to center lane, angle between center lane and vehicle	Acceleration, steering, brake	Speed, lane deviation, angle deviation, collisions, goal position
End-to-End Urban Driving by Imitating a Reinforcement Learning Coach	PPO	BEV Semantic Segmentation, steering, throttle, brake, gear, lateral and horizontal speed	Steering, throttle, brake	Collision, traffic light/sign, route deviation, being blocked, speed

Table 2 (cont.)

Source	Algorithm	States	Actions	Rewards
CARLA: An Open Urban Driving Simulator	A3C	Semantic Segmentation, measurement vector, speed, distance to goal, damage from collisions	Steering, throttle, brake, hand brake, reverse gear	Speed, distance traveled, collision, overlap with sidewalk and opposite lane
Safe Navigation: Training Autonomous Vehicles using Deep Reinforcement Learning in CARLA	DQN	Segmentation, depth images, obstacles, position, velocity proximity of other vehicles, measurement vector	Steer, throttle, brake	Speed, collisions, path following

Table 2 (cont.)

Source	Algorithm	States	Actions	Rewards
Reinforcement learning-based autonomous driving at intersections in CARLA Simulator	PPO	Distance to intersection, longitudinal velocity. State vector of individual states for each lane and ego vehicle	Velocity, stop and drive	Velocity, crossing intersection, collisions, episode duration

Simulators and Scenarios

When developing an autonomous vehicle utilizing reinforcement learning it is important to train the vehicle in a simulation. A vital reason for utilizing a simulator is safety. Using a simulator ensures there are no dangers by testing the vehicle in a real-world scenario before sufficient training is done in order to develop an adequate model. Utilizing a simulator also benefits via cost reduction. Simulator set ups are dramatically less expensive than real world testing scenarios. Time consumption is also decreased via simulations as a simulation can quickly be set up compared to setting up real world scenarios. The scalability in simulations are also greater than real-world scenarios. Simulations provide a variety of conditions when compared to real world scenarios. The ability to control the scenario in a simulator assists in developing robust models, which can then be transferred to real-world testing. The simulators commonly used for reinforcement learning applications involving autonomous vehicle include Car Learning to ACT (CARLA), AirSim and Simulation of Urban Mobility (SUMO). CARLA

features a variety of both urban and rural cities/towns. CARLA gives the user the ability to train in a variety of scenarios as CARLA provides a variety of options. CARLA provides a realistic 3D simulation with high quality graphics and physics. CARLA has a well-documented Python API, which allows for the spawning of other vehicles and pedestrians. The CARLA API also allows for simple camera and sensor implementation. Weather conditions can be modified in the CARLA simulator and custom maps can also be developed within CARLA. AirSim is commonly used for drone simulation but also provides an environment for autonomous vehicle simulation. Similar to CARLA, AirSim provides realist graphics and physics and customizable weather scenarios. AirSim also provides a well-documented API which can be used to simply the simulation setup. SUMO is visually less appealing compared to the two previously mentioned simulators. Rather than an immersive 3D environment, SUMO simplifies its graphics in order to focus more on traffic flow simulations. This strategy allows for SUMO to better simulate large-scale scenarios. Comparing the three simulators, it can be seen that CARLA and AirSim will lead to more immersive urban driving scenarios, where SUMO may be used for simplified graphics to handle large-scale driving scenarios. Resources including autonomous vehicle simulation utilizing reinforcement learning can be found below, including the simulator used as well as the training scenario.

Table 3: Simulator and scenario comparison

Source	Simulator	Scenario
Learning to drive in a day	Custom 3D environment using Unreal Engine	Lane following

Table 3 (cont.)

Source	Simulator	Scenario
Reinforcement learning-based autonomous driving at intersections in CARLA Simulator	CARLA, SUMO	Intersection Navigation
Deep Reinforcement learning based control for Autonomous Vehicles in CARLA	CARLA	Route Navigation
Lane Change Decision-Making through Deep Reinforcement Learning with Driver's Inputs	CARLA	Lane Changing
Evaluation of Deep Reinforcement Learning Algorithms for Autonomous Driving	AirSim	Lane Following/Navigation

CHAPTER III

METHODOLOGY

Proximal Policy Optimization

Proximal Policy Optimization (PPO) was introduced with the intent to improve on previously developed methods in order to develop a robust, data efficient method that is able to be scaled to large models (Schulman et al., 2017, p.1). PPO also aims to simplify the previously developed method known as trust region policy optimization (TRPO), which is known to be complicated and underperforms when noise or parameter sharing is involved. PPO is a policy gradient method, which means that the policy is optimized directly (Schulman et al., 2017, p.1). The fundamental concept of PPO is that it alternates between sampling data through interactions with the environment, then a surrogate objective function is enhanced via stochastic gradient ascent. Rather than one gradient update per sample data, PPO has an objective function that allows for multiple epochs of updates in minibatches (Schulman et al., 2017, p.1). The way policy gradient methods are implemented, is an estimator of a policy gradient is computed, and then input into a stochastic gradient ascent algorithm. An example of a gradient estimator, provided by (Schulman et al., 2017, p.2) is as follows:

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t$$

Where the stochastic policy is π_θ, the advantage function estimator is $\hat{A}_t$ and the expectation that provides the advantage over a finite batch of samples is $\hat{\mathbb{E}}_t$. Differentiating the objective allows for the estimated $\hat{g}$ to be determined, this is done with the following policy gradient loss equation provided by (Schulman et al., 2017, p.2):

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[log\pi_\theta(a_t|s_t)\hat{A}_t$$

This method includes multiple optimization steps on the policy gradient loss with the same trajectory typically leads to large policy updates, which are commonly destructive to the policy. Rather than using long term expected rewards, PPO introduces a much simpler method to the objective known as the "surrogate" objective function. This method is meant to simplify the optimization process and assists in improving the policy with a gradient ascent method. The clipped surrogate objective utilized in PPO in order to improve the stability and efficiency of policy updates via a policy gradient method, given by (Schulman et al., 2017, p.3), is the following equation:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t\right)]$$

Here, ϵ is a hyperparameter. The conservative policy iteration objective (L^{CPI}) which is expressed with the following equation:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t]$$

would typically lead to excessively large policy updates without constraints (Schulman et al., 2017, p.3). Therefore, it is used as the first term inside the min. Modifications to the surrogate objective are done by clipping the probability ratio with the second term $clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t$. PPO takes the minimum of both the unclipped and clipped objective in order for the final objective to be a lower bound of the unclipped objective (Schulman et al., 2017, p.3). This method ignores changes in the probability ratio when objective improvements would occur and only includes it when the objective would deteriorate. This strategy assists in selectively ignoring changes in the probability ratio in order to maintain stable policy updates (Schulman et al., 2017, p.3).

Actor-Critic Architecture

In reinforcement learning, there are algorithms which include an actor-critic framework. A framework for actor-critic proximal policy optimization, provided by (Lim et al., 2020, p.7), can be seen below:

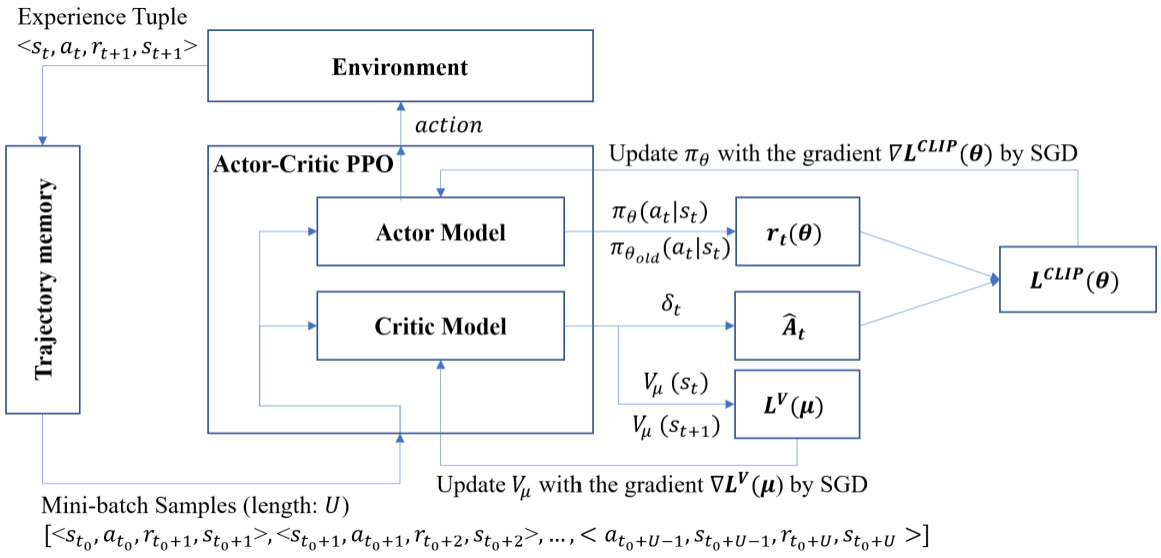

Figure 8: Actor-critic PPO framework

The important take-away from the actor-critic structure is the roles that both the actor and critic play in an actor-critic algorithm. The actor model is responsible for selecting actions for the agent while the critic critiques the actions. The important factor here is the critic component. The critic evaluates the action taken and determines how the action affected the state of the environment. The critic is used to provide feedback to the actor on how good or bad the chosen action was. Within the actor-critic PPO algorithm, a generalized advantage estimator (GAE) is used to calculate the estimator of the advantage function with the equation provided by (Lim et al., 2020, p.7):

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1}^V + (\gamma\lambda)^2 \delta_{t+2}^V \dots (\gamma\lambda)^{U-t+1}\delta_{U-1}^V$$

Where γ is the discount factor, the GAE parameter, λ, is ($\lambda \epsilon [0,1]$), U is the size of the minibatch sampled, and the temporal difference is provided by (Lim et al., 2020, p.7):

$$\delta_t = r_t + \gamma V_\mu(s_{t+1}) - V_\mu(s_t)$$

where V_μ estimates the expected return. These equations will play a key role for future considerations in a proposed conceptual framework.

Human Emotion

Human involvement in reinforcement learning is known as Human in the Loop reinforcement learning (HRL). This methodology has been implemented by (Bradley Knox et al., 2008), with training an agent manually via evaluative reinforcement (TAMER). Another implementation of human involvement in reinforcement learning is that of (MacGlashan et al., 2017), with convergent actor-critic by humans (COACH). These methods posed different approaches to include human involvement in the reinforcement learning algorithm. These works have been expanded on but pose as baseline models for including human emotion. Therefore, these methods are considered when researching implementation strategies. A general feedback loop including human feedback, given by (Poole et al., 2024, p.10), can be seen below:

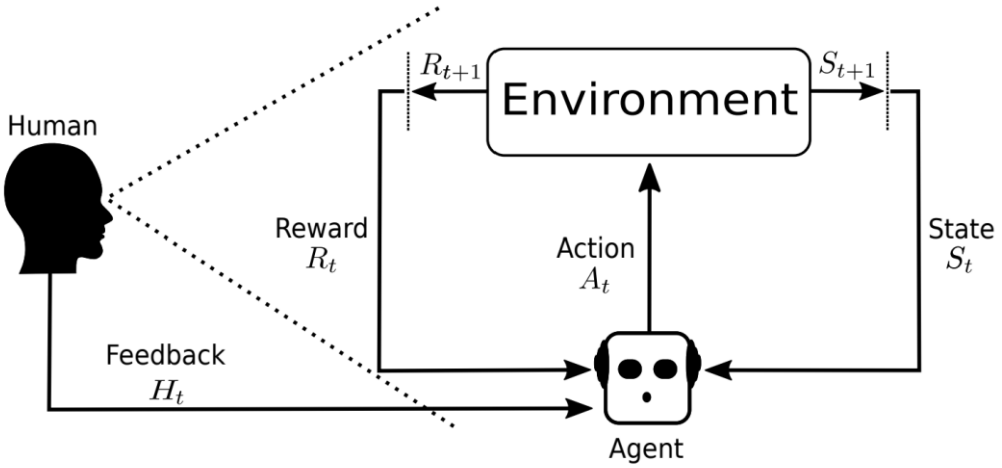

Figure 9: General feedback-based setup

The advancements in autonomous vehicles may one day lead to the commercialization of autonomous vehicles. Consumers may use these as personal vehicles, while companies may deploy these vehicles as taxi services. One thing is certain, if autonomous vehicles are commercialized, humans will become passengers in an autonomous vehicle. It will be important that these autonomous vehicles learn to consider the emotions of their human passengers. This may assist in more human like driving behavior, but more importantly, contribute to the comfort of the passenger during their experience as a passenger. Therefore, a conceptual framework is proposed to include human facial emotion into the reinforcement learning algorithm for autonomous vehicles. There are a few factors to consider when including facial emotion as a form of feedback. These considerations for the conceptual framework can be seen in the following modified diagram taken from (Poole et al., 2024, p.10), where the green text boxes represent the chosen strategy:

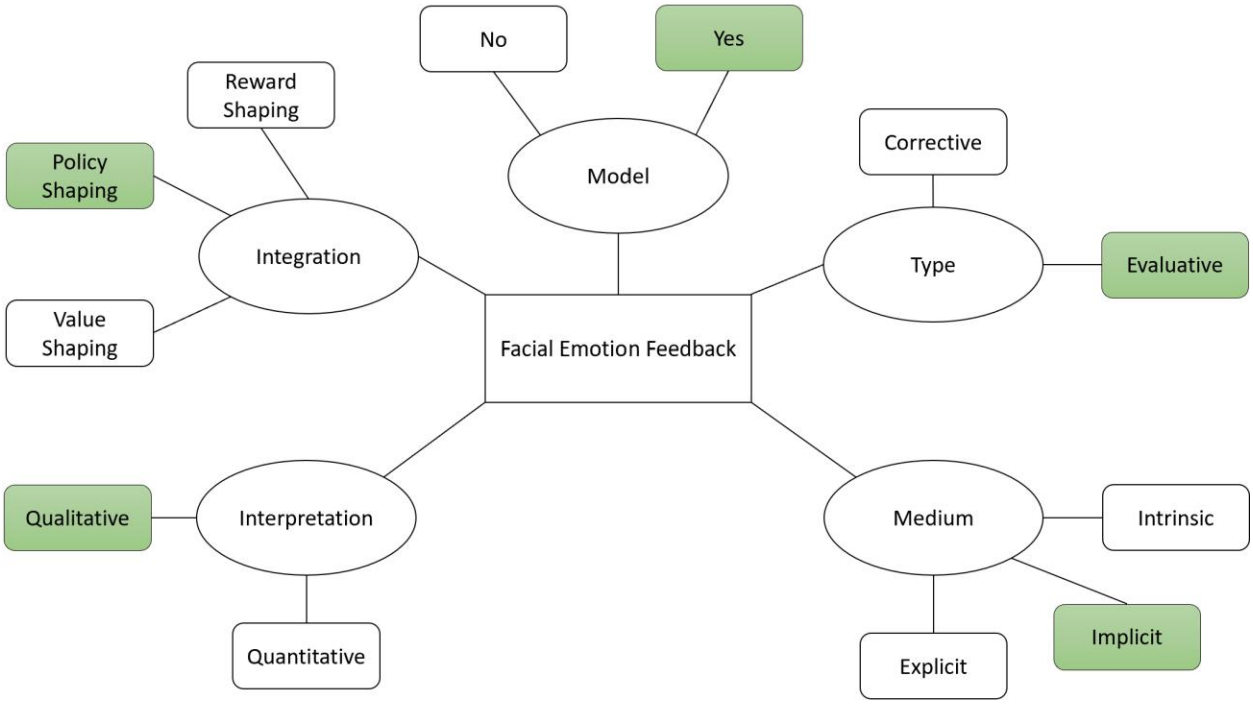

Figure 10: Conceptual framework diagram

The conceptual framework will involve including human facial emotion to an existing reinforcement learning framework in such a way it may be applicable to autonomous vehicle research. Considerations included in the human facial emotion feedback include type of feedback, the medium, interpretation, integration and modeling. The intention of the conceptual framework is that the feedback will be implemented into an actor-critic algorithm due to its best fit methodology. In terms of the type of feedback, facial emotion will be considered to be evaluative feedback. This is due to the nature of facial emotions. Facial emotions provide a description of how a person feels in response to their environment. Facial emotions do not give explicit instructions on what should change, rather it is used to assess a situation. Corrective feedback directly points out problems with explicit feedback, whereas evaluative feedback gives guidance. This evaluative feedback is crucial, as it will allow for the reinforcement learning

method to maintain its learning-based approach made via interactions perceived through facial emotions. This type of evaluative feedback contributes to the medium of this feedback being implicit. Rather than giving direct verbal-cues which would be deemed an explicit medium, the nature of facial expressions is that they act as subtle cues and provide indirect reactions. Aside from direct verbal commands, another form of explicit response, such as the method used in (Bradley Knox et al., 2008), include button pushing methods. This method involved pushing a positive reward button and a negative reward button, resulting in a reward shaping method. The reason implicit response is considered, is because this methodology enables the agent to become more attuned to the comfort levels of the human passenger, by helping refine and adjust the decision-making process, rather than determining the decision-making process. In terms of the interpretation of the conceptual framework, there will be a need for qualitative-quantitative mapping. Facial emotion as feedback is provided via a qualitative nature. That is the raw data, provided as facial expression, are categorized via a convolutional neural network (CNN). However, the algorithms used in reinforcement learning require quantitative data. Therefore, a mapping from qualitative, e.g. the classification process of facial emotions, will need to be mapped to a quantitative interpretation e.g. values assigned to facial emotions. This can be done by providing positive values to facial emotion such as happy or neutral, and assigning negative values to emotions such as fear. This methodology allows for seamless integration into the reinforcement learning algorithm. The quantitative feedback derived from the qualitative data will assist in supporting the policy shaping approach based on the integrated feedback. Policy shaping integration will be incorporated by influencing the policy indirectly. This is done by including the facial emotion data into the critic component of an actor-critic model. Rather than directly modifying the reward structure, the facial emotion feedback will influence the policy via

the value function of the critic. The policy will then be shaped over time via the continuous feedback.

Including Human Emotion

There are a few considerations when including the facial emotion as intended. Two contributing equations, from (Lim et al., 2020, p.7), included in the actor-critic PPO algorithm are considered. The first equation is the generalized advantage estimator (GAE), used to calculate the estimator of the advantage function, given by the following equation:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1}^V + (\gamma\lambda)^2 \delta_{t+2}^V \dots (\gamma\lambda)^{U-t+1}\delta_{U-1}^V$$

The second equation is the temporal difference equation given as:

$$\delta_t = r_t + \gamma V_\mu(s_{t+1}) - V_\mu(s_t)$$

where V_μ estimates the expected return. In theory, including human facial emotion into either of these equations will keep the human emotion in the critic component of the actor-critic algorithm to assist in policy shaping. Let us denote human reward as r_h. Including the $r_h$into the temporal difference, δ_t, equation will result in r_h directly adjusting the immediate state transitions. As a result, immediate behavior changes may occur in order to align the behavior of the agent with human facial emotion. Including r_h into the generalize advantage estimator, $\hat{A}_t$, will result in r_h affecting future decisions, which will be reflected in policy updates. There are a number of considerations when including r_h into either of the equations proposed. These considerations include whether r_h will have an immediate impact, or an impact over time. The stability and sensitivity are also considered, as including r_h into δ_t may result in a lack of stability due to sudden shifts in emotional states. Theoretically, by including r_h into $\hat{A}_t$ will result in a balanced policy adjustment, due to the emotions being integrated over time. There will need to be some

experimentation on the impacts of including human emotion. A consideration is to include a weight that can be associated with human reward to fine tune the influence that r_h will have.

CHAPTER IV

MODEL IMPLEMENTATION AND RESULTS

CARLA Simulator

The CARLA simulator plays a crucial role in model implementation. CARLA serves as the environment for this reinforcement learning application dealing with an autonomous vehicle. CARLA is chosen as a simulator due to its realistic graphics and physics. These properties contribute to an immersive simulation crucial for end goal testing scenarios. The town chosen for the simulation is town 2. Town 2 is a small town with a number of junctions to create unique experiences for the vehicle. There are stretches of road with multiple junctions and a long stretch of road without any junctions.

CARLA API provides waypoints and allows for the waypoints to be connected in order to generate a path for the vehicle to follow. An example of how waypoints are implemented in the CARLA environment, provided by (Razak, 2022, p.34), can be seen below:

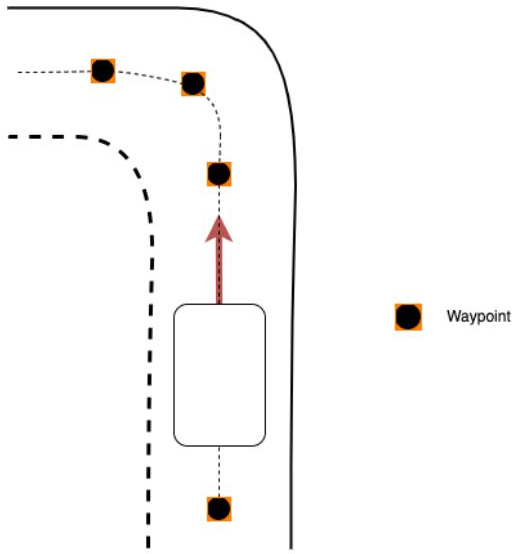

Figure 11: Waypoint information

The vehicle is meant to navigate the outer parameter of the town, for a total of 780m, without entering the center of the town. The path the vehicle is meant to follow, highlighted in blue, provided by (Razak, 2022, p.33), can be seen below:

Figure 12: Waypoint pathing

This method simplifies the path the vehicle has to take, but still provides multiple situations where junctions in the road must be experienced. CARLA allows for both pedestrians and vehicles to be spawned into its environment in order to add variance to the simulation. CARLA assists in simplifying the perception of the vehicle by providing API to include cameras and sensors. The simulation utilizes a front facing semantic segmentation obtained via a semantic segmentation camera provided via CARLA API. The dimensionality of the image is reduced with the implementation of a CNN-based variational autoencoder (VAE). Additional sensors that are crucial to the simulation include a collision sensor and lane invasion sensor.

Baseline Model

An existing GitHub repository, developed by (Razak, 2022) was utilized in this experiment. The work of (Razak, 2022) is strongly based on the work of previously mentioned (Kendal et al., 2019), Learning to Drive in a Day. The repository provided by, (Razak, 2022), shares a number of contributions including training a VAE on semantically segmented images rather than an RGB camera. The implementation of (Razak, 2022), plays a key role, as it serves as a baseline model to be tested and modified. The goal in mind when modifying the existing repository is to implement modifications that would prove beneficial for obtaining realistic reactions from passengers partaking in the simulation. PPO is used by (Razak, 2022), which is an actor-critic framework. This methodology is key for future implementation of the conceptual framework proposed to include human facial emotion. It is important to investigate the methods used to implement the model developed by (Razak, 2022). Therefore, the observations, actions and rewards will be investigated. The observations included in this model, is an encoded front camera image, throttle, velocity, previous steer, distance from the center of the lane and angle deviation from the center lane in the observation space (Razak, 2022). These observations are important as they provide vital information to the agent and are directly tied to the reward function being implemented. Two important factors from the observation space that may need clarification, include the deviation from the center lane and angle deviation from the center lane. An illustration used to assist in the explanation of these observations, provided by (Razak, 2022, p.35), can be seen below:

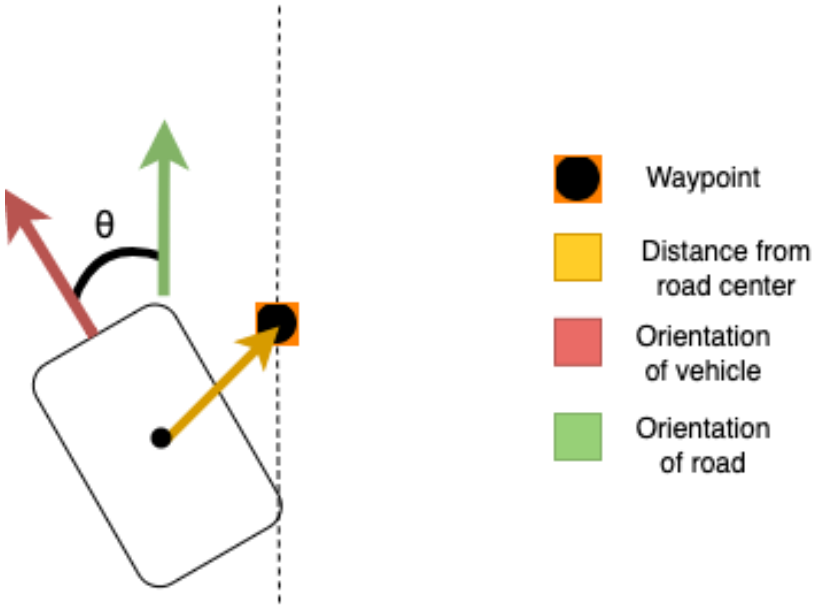

Figure 13: Observation of orientation representation

The distance from the center of the road includes the trajectory of the waypoint, creating a line down the center of the road. The distance from the center of the vehicle to the center of the road is calculated and used as the distance from the center of the road observation. The orientation of the vehicle with respect to the orientation of the road is also used to develop an angle of deviation for the forward vector of the vehicle, with respect to the forward vector of the road created with the waypoint. The intent of these two factors is to keep the vehicle within the boundary of the lane, while assisting in keeping the vehicle in line with the direction of the road. The actions provided to the agent are steering [-1,1] and throttle [0,1]. In terms of rewards, it may be beneficial to start with terminating factors, which terminate an episode and provide negative rewards to the agent when met. These terminating factors include collisions, exceeding a max distance from the center lane, remaining stopped for the first ten seconds of an episode, and exceeding a velocity determined by a declared max speed. These terminating factors assist in avoiding collisions, staying within the intended lane, learning to move forward and learning not

to speed. Each of these terminating factors not only end the episode, but also return a reward of negative ten (Razak, 2022, p.40). Next are the rewards termed centering factor and angle factor. These rewards are important as they contribute to the reward that is calculated at every timestep. The centering factor reward is a linear reward from zero to one, where the maximum reward is achieved when the vehicles lane deviation is zero and returns a reward of zero when the maximum distance from the center, in this case 3m, is reached. Similarly, the angle factor reward is also distributed linearly, from zero to 1. This reward regards the angle of deviation previously mentioned, where a reward of 1 is given for an angle deviation of 0° and returns a reward of 0 when angle deviation reaches 30°. The visualization of the centering factor reward and angle factor reward can be seen below:

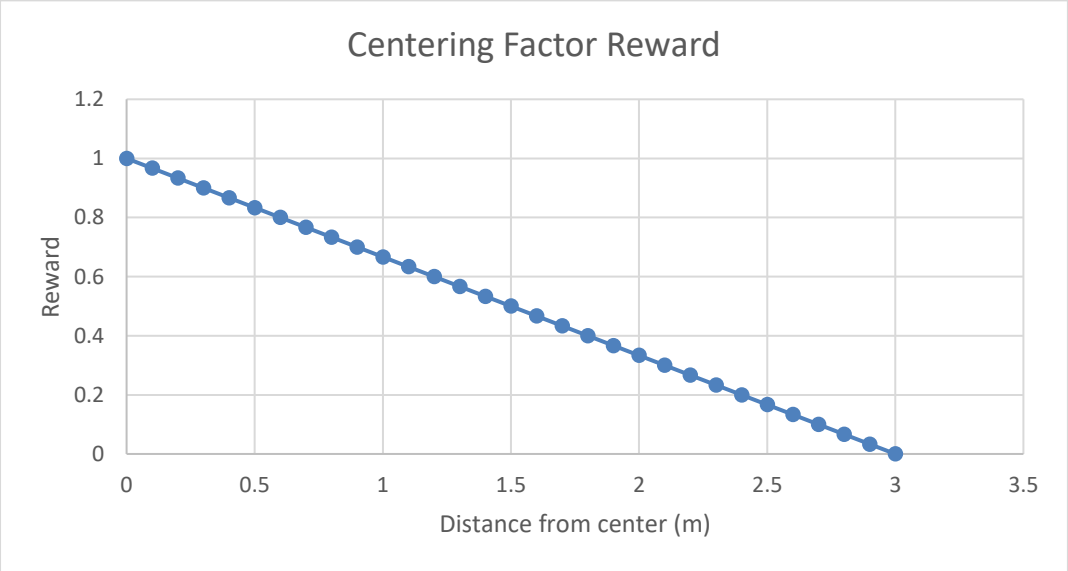

Figure 14: Centering factor reward

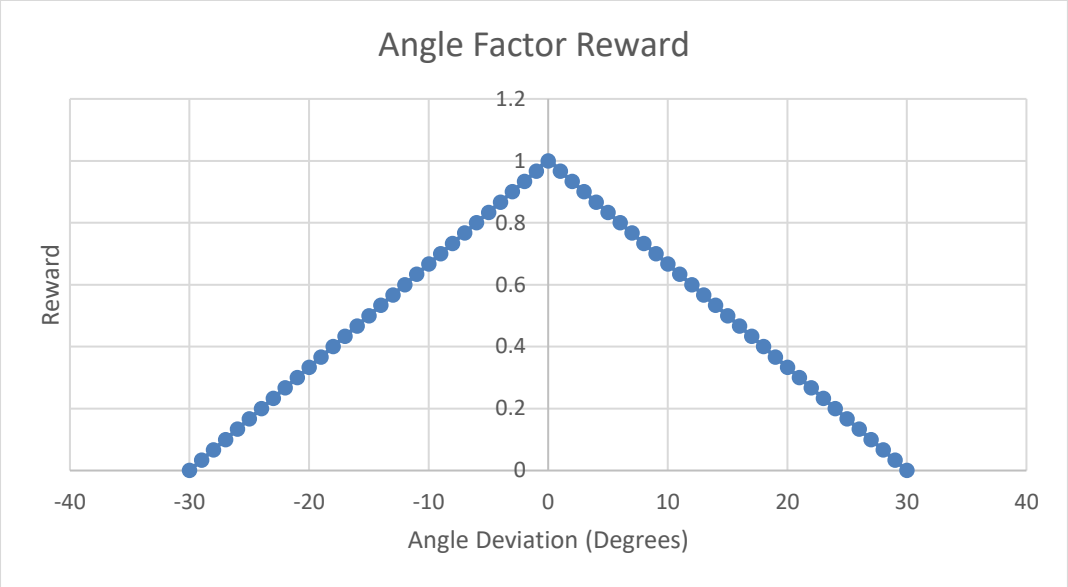

Figure 15: Angle factor reward

The rewards given for speed are also given linearly. Note that values of target speed, minimum speed, and maximum speed are given by (Razak, 2022) as 22 (km/h), 15 (km/h) and 25 (km/h) respectively. There are two direct considerations when attributing reward to speed. These cases include when the velocity is less than the minimum speed and when velocity is greater than the target speed. When the velocity of the vehicle is less than the minimum, reward is given linearly from zero to one, for speeds of 0 to 14 (km/h). When the velocity is greater than the target speed, the reward starts at zero for 23 (km/h) and linearly approach a value of -0.33 for when speed reaches 24 (km/h). The rewards for these two criteria can be seen below:

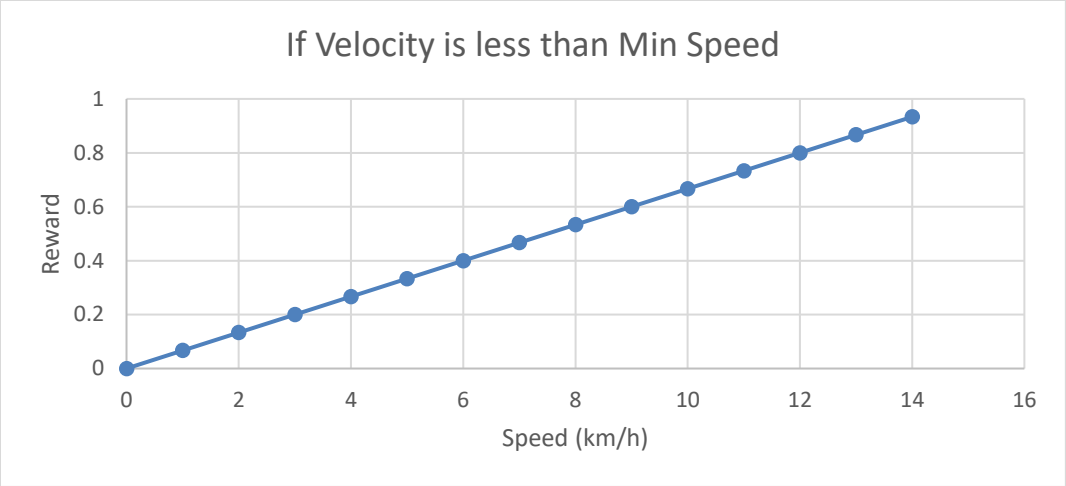

Figure 16: Velocity reward for velocity being less that minimum speed

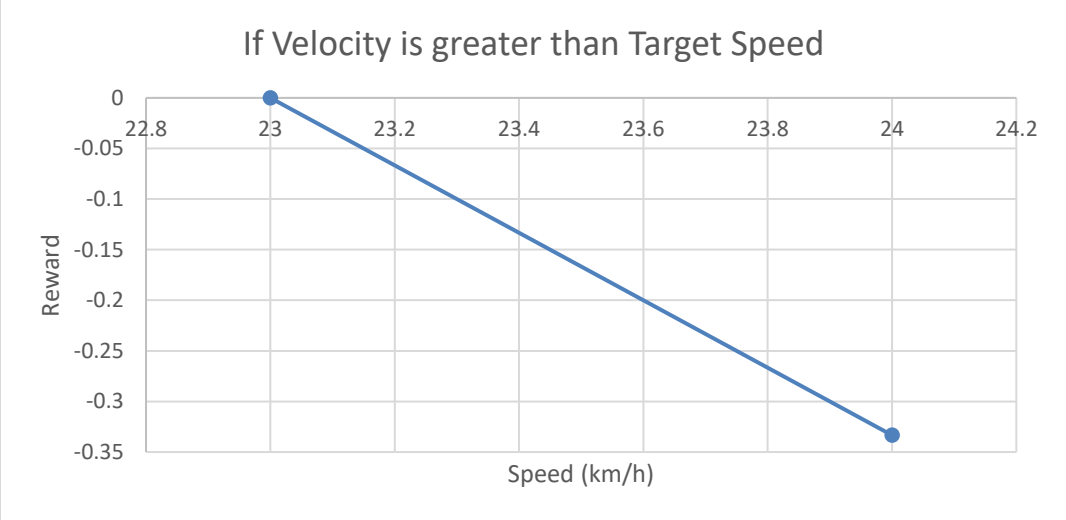

Figure 17: Reward for velocity greater than target speed

If neither of these forementioned criteria are met, the reward for speed is given as 1 as it is deemed that a vehicle speed within the range of 15 (km/h) to 23 (km/h) are ideal. These speed rewards are then multiplied with the centering factor and angle factor to give a reward for each individual timestep, that are then summed for an overall reward for an episode (Razak, 2022, p.39).

Modified Simulation and Models

The base model provided by (Razak, 2022) needed to be retrained, as it was not possible to run the trained agent by simply pulling the repository and running a trained agent as instructed in the repository. To begin, the VAE was retrained in the same method as described by (Razak, 2022). No parameters were changed, but new files were created to take place of the original files associated with the VAE after retraining the VAE, as these files did not seem to be working as intended. Checkpoint and pretrained model files were then replaced with new files from training the model from scratch. The training scenario was implemented in the same manner as described by (Razak, 2022) and similar results were able to be obtained. The first tests were implemented by running a series of 100 episodes on the base model. Two tests were run, one test including zero vehicles in the environment with the autonomous vehicle, and one test with 40 vehicles in the testing scenario. In order to implement these tests, a different method was used to include other vehicles in the simulation. A method for including other vehicles is provided by (Razak, 2022), however, when implementing his method issues occurred. To better fit the implementation of other vehicles, the vehicles are added to their own unique list. These vehicles are then called upon at the start of each episode and then destroyed at the end of every episode. This was a methodical approach, as spawning vehicles at the beginning of the simulation and leaving the vehicles in the simulation would cause issues, such as the vehicles getting stuck and then blocking sections of the road. The implementation of other vehicles allowed for the base model to be compared for scenarios where other vehicles were not included in the simulation and where other vehicles were included in the simulation. Another consideration to drawing out human emotions is including dynamic speed limits to the simulation. CARLA provides a method for providing boundary boxes for actors in the simulations such as traffic lights and traffic signs.

In the scenario created to include dynamic speed limits, only a few specific speed limit signs were considered to simplify the experiment. First, in order to include dynamic speeds, the reward function would have to be modified. The reward function included in the base model provided by (Razak, 2022) includes a minimum speed, target speed and maximum speed. Due to these declared variables: max speed and minimum speed, implementing dynamic speeds would be more complex. Therefore, a similar method for speed reward was implemented and the model would be retrained. Rather than using minimum speed and maximum speed, a speed reward function was developed with only target speed as a contributing factor. This method would allow for the target speed to change and dynamically change the rewards. The speed reward was given linearly from a value of 0 to 1, for speeds of 0 km/h to the target speed. The reward would then drop linearly from 1 to 0 from target speed to a speed greater than 5 km/h of the target speed. The newly implemented rewards for speed can be visualized below:

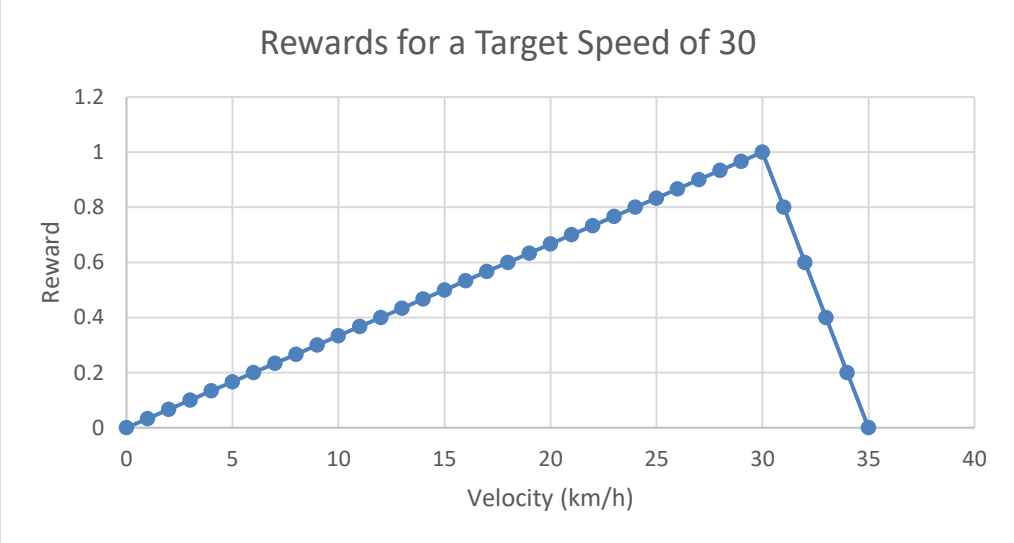

Figure 18: Modified reward for a target speed of 30 (km/h)

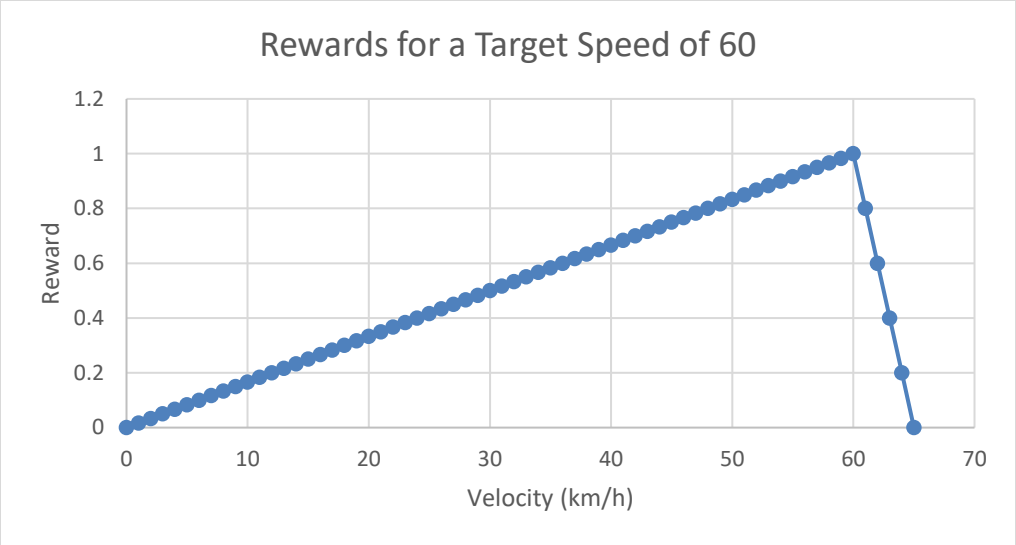

Figure 19: Modified reward for a target speed of 60 (km/h)

Once the changes to the speed rewards were made, the agent was able to be retrained. Results from the newly implemented rewards show that the agent performed just as well as the originally implemented base model. A custom script was then included to inform the autonomous vehicle of speed limit changes alternating between 30 km/h and 60 km/h. The decision to only include two dynamic speeds limits was made to simplify the experiment. This decision was made due to there being speeds reaching 90 km/h via speed limit signs, and sections of road that quickly change from 30 km/h to 60 km/h and back to 30 km/h in a short section of road. Once dynamic speed limits were implemented into the environment, the newly developed model with only target speed being used in the reward structure was tested. The purpose of including the dynamic speeds in this testing process, is to understand if the trained model would handle the dynamic speed limits as intended. It is clear that the actions of the agent limit the behavior of the vehicle, as only steering and throttle actions are available to the agent. It is important that brakes are added to the vehicle as an action. Adding brakes would add another action to the agent. If added individually in such a way where, steering [-1,1], throttle [0,1] and

brake [0,1] are each an individual action, it would further complicate the agents training. If brakes were to be implemented in this way, as the agent begins to take actions, it will most likely choose values for both throttle and brake. In the CARLA simulator, when a vehicle is at rest, a value of brake will always override a value of throttle, preventing forward movement of the vehicle. A method, found to be successful by (Zhang et al., 2021) is to merge throttle and brake into one action. Rather than using throttle and brake in the action space, acceleration can be used as a second action to steering. In this case, acceleration is set to [-1,1], then positive acceleration is mapped to the throttle control and negative acceleration is mapped to the brake control. This method not only simplifies the action space but also provides more realistic driving controls, as it is not common for both throttle and brake to be used in typical driving scenarios. Adding brakes as an action allowed for a model to be trained with the ability to brake as an action. The agent which now included braking as an ability was trained, utilizing the same base model developed by (Razak, 2022), the only factor that was changed was the ability to brake.

Results

The first results presented include the results from the base model which only include steering and throttle as actions. The results show the average reward obtained by the model over a series of 100 episodes. The results are illustrated as a bar graph, where one bar represents the average reward over 100 episodes with no vehicles in the simulations, and the other bar represents the average reward over 100 episodes with 40 vehicles in the simulation. These results can be seen below:

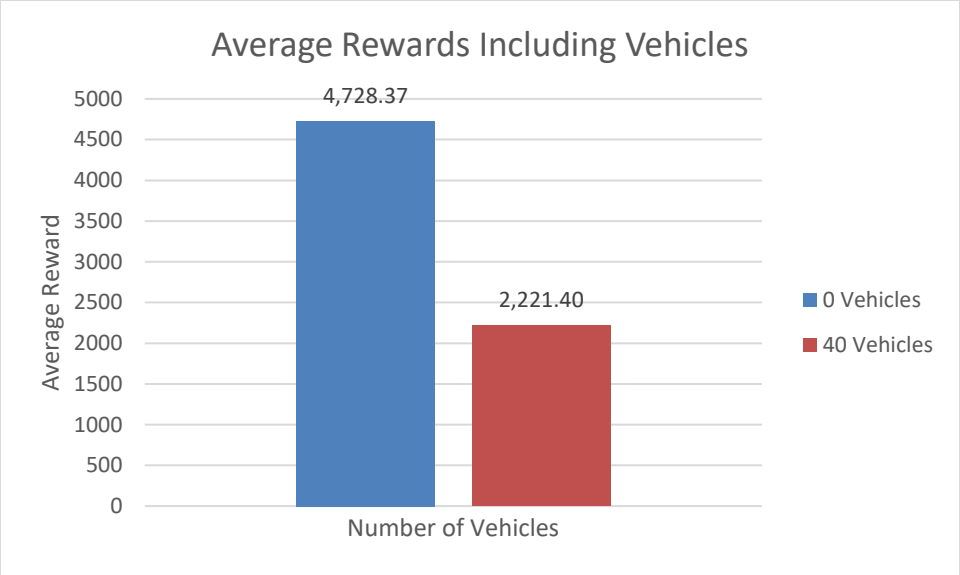

Figure 20: Average reward when including other vehicles

It is clear to see that by adding vehicles to the simulation the vehicle under performs. The average reward for the vehicle with zero vehicles in the simulation is 4728.37 while the average reward for the vehicle with 40 vehicles in the simulation is 2221.40. This is a result of the vehicle not being trained to handle other vehicles in the environment. Even though the vehicle is trained to avoid collisions, it does not have enough information to handle other vehicles within the environment. The next set of results includes the behavior of the vehicle trained with the new reward structure for speed that only includes target speed. The vehicle was trained on the new rewards, then tested to see if the model could handle the dynamic speed limits. To illustrate the target speed was printed and the deviation from the target speed was collected. The results can be seen in a bar graph shown below:

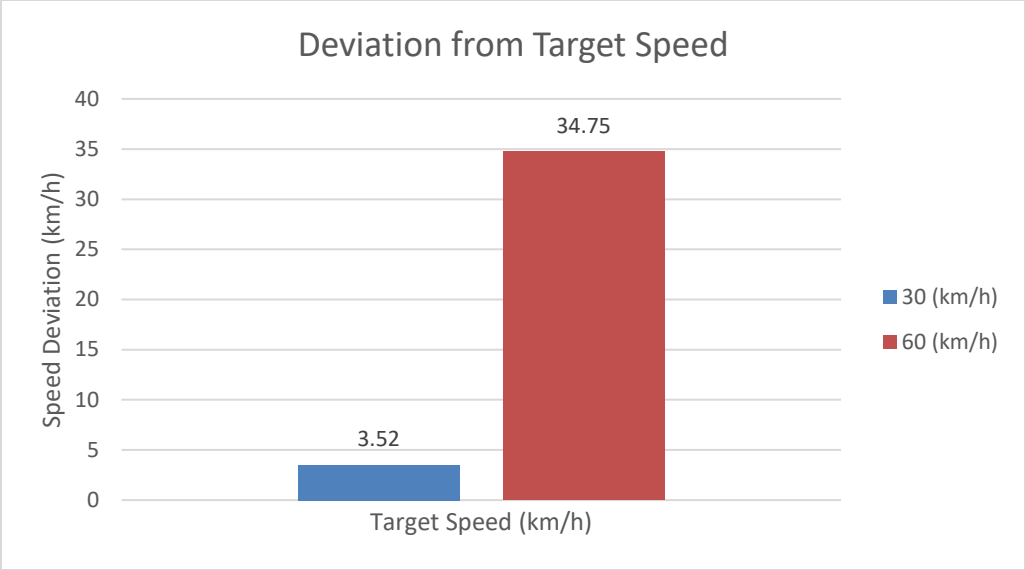

Figure 21: Deviation from target speed

It can be seen that the vehicle does learn to stay within an acceptable range (3.52 km/h) of the target speed when the target speed if 30 (km/h). However, it can also be seen that the vehicle struggles to stay within an acceptable range (34.75 km/h) of the target speed when the target speed is 60 (km/h). This may be a result of the model not being directly trained with dynamic target speeds. The model is trained on maintaining a target speed, however, due to the dynamic changes in the reward system, due to the alternating target speed, the model may not have enough information to handle the dynamic speeds in the environment during testing. Results from training a model with brakes included in the action space were documented. The only change to the model was the addition of brakes, and the time determining whether the vehicle was making forward progress or not was changed from 10 seconds to 20 seconds. The results of minimum training on the agent can be seen below:

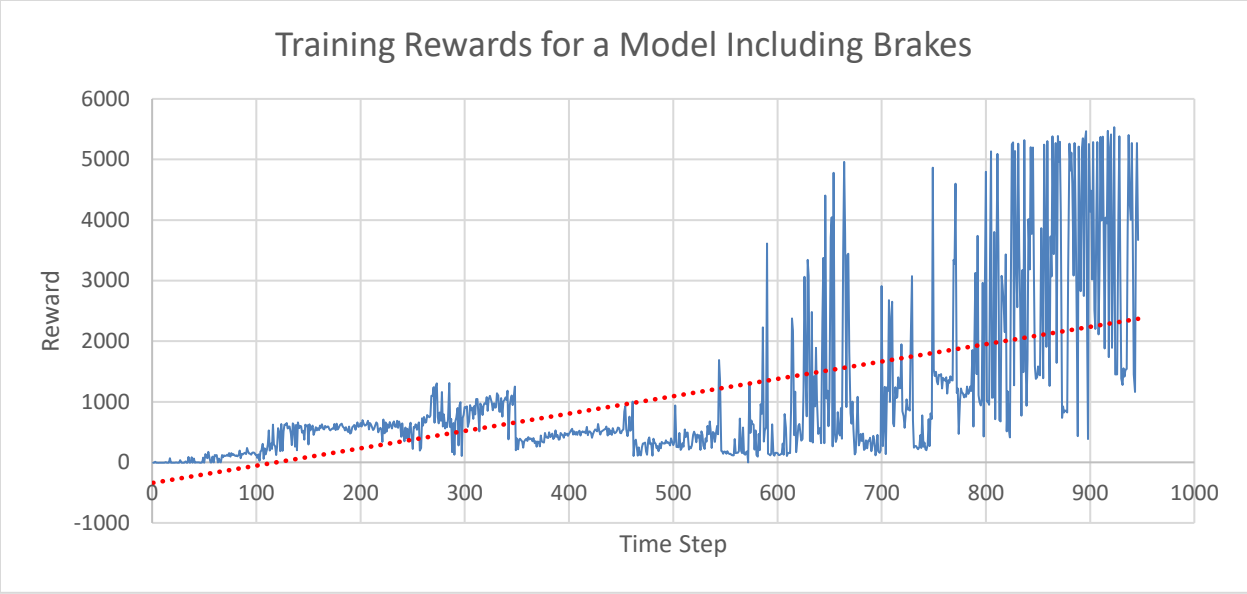

Figure 22: Training rewards for braking model

The graph represents the minimal training needed to show that the model is converging to acceptable performance. There is some noise to the graph but it can be seen that the overall trend is an increase in performance. The model begins to achieve rewards comparable to the maximum rewards received by the base model without the ability to brake. To further ensure that the agent is using both throttle and brake as actions, the number of times throttle was greater than zero and brake was greater than zero were tracked throughout an episode during training. The results of the number of times either throttle or brake were used during an episode can be seen below:

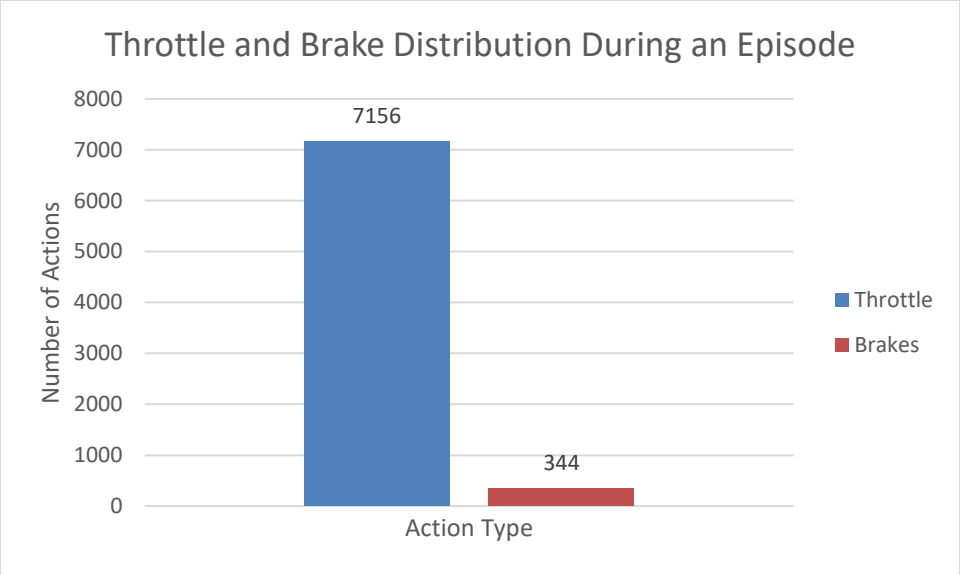

Figure 23: Throttle and brake distribution

The results show that the agent utilized throttle 7156 times and brake 344 times during an episode. This is important to note because even though the agent is deep into training, without any other vehicles or dynamic speeds, the agent still chooses to use both the throttle and brake controls provided by the acceleration action.

CHAPTER V

CONCLUSION

Presented in this work is a conceptual framework for including human facial emotion into a reinforcement learning approach for autonomous vehicles. The development of an autonomous vehicle using PPO is put into practice. A baseline model is trained and tested on performance in different scenarios. The performance of the baseline model is not sufficient for intended application; therefore, modifications are made in an attempt to produce a worthy model. Considerations include adding vehicles and dynamic speeds to the simulations. A method for including other vehicles into the simulation is implemented and tested. Dynamic speed limits are added to the simulation and tested. The addition of brakes to the vehicle control via the action space is implemented. The model is trained and the results from training are gathered. Results show that more work is needed for an adequate model that will be required for future work of the conceptual framework, that is including human facial emotion.

Future Work

Future work will include developing an autonomous vehicle that meets the standard required to include human facial emotion. The vehicle will need further modifications in order to handle other vehicles and dynamic speed limits. Once a model is developed to handle these scenarios, the conceptual framework to include human facial emotion may be put into practice.

REFERENCES

Aradi, S. (2022). Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems, 23*(1), 740-759. https://doi.org/10.1109/TITS.2020.3024655S

Berntorp, K. (2017). Path planning and integrated collision avoidance for autonomous vehicles. *2017 American Control Conference (ACC)*. https://doi.org/10.23919/acc.2017.7963572

Bradley Knox, W., & Stone, P. (2008). Tamer: Training an agent manually via evaluative reinforcement. *2008 7th IEEE International Conference on Development and Learning*. https://doi.org/10.1109/devlrn.2008.4640845

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017, October). CARLA: An open urban driving simulator. In *Conference on robot learning* (pp. 1-16). PMLR.

Folkers, A., Rick, M., & Buskens, C. (2019). Controlling an autonomous vehicle with deep reinforcement learning. *2019 IEEE Intelligent Vehicles Symposium (IV)*. https://doi.org/10.1109/ivs.2019.8814124

Gutiérrez-Moreno, R., Barea, R., López-Guillén, E., Araluce, J., & Bergasa, L. M. (2022). Reinforcement learning-based autonomous driving at intersections in CARLA simulator. *Sensors, 22*(21), 8373. https://doi.org/10.3390/s22218373

Ignatious, H. A., Sayed, H.-E.-, & Khan, M. (2022). An overview of sensors in autonomous vehicles. *Procedia Computer Science, 198*, 736–741. https://doi.org/10.1016/j.procs.2021.12.315

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., & Shah, A. (2019). Learning to drive in a day. *2019 International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/icra.2019.8793742

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A., Yogamani, S., & Perez, P. (2022). Deep reinforcement learning for autonomous driving: a survey. *IEEE Transactions on Intelligent Transportation Systems, 23*(6), 4909–4926. https://doi.org/10.1109/tits.2021.3054625

Kuutti, S., Bowden, R., Jin, Y., Barber, P., & Fallah, S. (2019). A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2019.2962338

Lim, H.-K., Kim, J.-B., Heo, J.-S., & Han, Y.-H. (2020). Federated reinforcement learning for training control policies on multiple IOT devices. *Sensors*, *20*(5), 1359. https://doi.org/10.3390/s20051359

Liu, C., Lee, S., Varnhagen, S., & Tseng, H. E. (2017). Path planning for autonomous vehicles using model predictive control. *2017 IEEE Intelligent Vehicles Symposium (IV)*. https://doi.org/10.1109/ivs.2017.7995716

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Wang, G., Roberts, D. L., ... & Littman, M. L. (2017, July). Interactive learning from policy-dependent human feedback. In *International conference on machine learning* (pp. 2285-2294). PMLR.

Nehme, G., & Deo, T. Y. (2023). Safe Navigation: Training Autonomous Vehicles using Deep Reinforcement Learning in CARLA. *arXiv preprint arXiv:2311.10735*.

Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L. M., Gómez-Huélamo, C., Gutiérrez, R., & Díaz-Díaz, A. (2022). Deep reinforcement learning based control for autonomous vehicles in CARLA. *Multimedia Tools and Applications*, *81*(3), 3553–3576. https://doi.org/10.1007/s11042-021-11437-3

Poole, B., & Lee, M. (2024). Towards interactive reinforcement learning with intrinsic feedback. *Neurocomputing*, 127628.

Razak, A.I. (2022). Implementing a deep reinforcement learning model for autonomous driving [Master's thesis, Budapest University of Technology and Economics]. GitHub. Retrieved from https://github.com/idreesshaikh/Autonomous-Driving-in-Carla-using-Deep-Reinforcement-Learning/blob/main/info/documentation/%5BThesis%202022%5D%20IMPLEMENTING%20A%20DEEP%20REINFORCEMENT%20LEARNING%20MODEL%20FOR%20AUTONOMOUS%20DRIVING.pdf

Sutton, R. S., Bach, F., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press Ltd.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Stang, M., Grimm, D., Gaiser, M., & Sax, E. (2020). Evaluation of deep reinforcement learning algorithms for autonomous driving. *2020 IEEE Intelligent Vehicles Symposium (IV)*. https://doi.org/10.1109/iv47402.2020.9304792

Wu, Y., Yin, Z., Yu, J., & Zhang, M. (2022). Lane change decision-making through deep reinforcement learning with driver's inputs. *2022 IEEE 7th International Conference on Intelligent Transportation Engineering (ICITE)*. https://doi.org/10.1109/icite56321.2022.10101421

You, C., Lu, J., Filev, D., & Tsiotras, P. (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems, 114,* 1–18. https://doi.org/10.1016/j.robot.2019.01.003

Zhang, Z., Liniger, A., Dai, D., Yu, F., & Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv48922.2021.0149

VITA

Timothy Marion Jude Lyons moved to Texas in 2010 where he soon obtained his associate's degree in engineering science at South Texas College. He then earned his bachelor's degree in mechanical engineering, in May 2021, from the University of Texas Rio Grande Valley with Magna Cum Laude. He began his graduate studies in August 2021 at the National Science Foundation CREST Center for Multidisciplinary Research Excellence in Cyber-physical Infrastructure Systems (MECIS). His research was funded by the National Science Foundation. He obtained his Masters of Science in Mechanical Engineering from The University of Texas Rio Grande Valley in August 2024. He can be reached at lyonsracing019@gmail.com.