AUTONOMOUS CONTROL OF AN ALL-TERRAIN VEHICLE USING EMBEDDED SYSTEMS AND ARTIFICIAL INTELLIGENCE TECHNIQUES

by

Karim H. Erian

A Dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2022

Approved by:

_____
Dr. James M. Conrad

_____
Dr. Dipankar Maity

_____
Dr. Chen Chen

_____
Dr. Aidan Browne

ABSTRACT

KARIM H. ERIAN. Autonomous Control of an All-terrain Vehicle using Embedded Systems and Artificial Intelligence Techniques. (Under the direction of DR. JAMES M. CONRAD)

The need for autonomous vehicles that do not operate on highways and can move off-road is increasing. While there have been significant advancements in autonomous passenger vehicles in the last couple of years, autonomous off-road vehicles have not received as much of this attention. The demand to deliver emergency supplies to areas unreachable by typical highways advocates the need for an off-road self-driven vehicle.

This dissertation discusses transforming an All-Terrain Vehicle into a self-driven off-road vehicle that can follow a path in a forest environment. The approach is to install actuators, sensors, microcontrollers, and a graphical processing unit on top of the ATV without changing the initial ATV architecture.

This research was composed of two phases. The first phase was to augment the ATV with actuators and sensors to have a reliable base where digital signals control the ATV. This phase focused on controlling the handlebar of the ATV in a forward direction, having a feedback speed control system, and implementing a braking control system. The second phase involved developing an Artificial Intelligence model with an image processing system to compute the proper steering angle and control the ATV. This phase consisted of an OpenCV module to read a camera feed and pass it to a semantic segmentation deep learning model called SegNet, which helps identify the paved path. The software received the semantic segmentation output data identified by the ATV's current position on the paved path and the expected trajectory of the ATV. A trained Machine Learning Linear Regression model used this data to predict the handlebar angle, pass the angle to a CAN bus to steer the ATV, and keep the ATV on the paved path. The dissertation includes the data gathering process, the

ML model training, and the LR model choice.

The research also included a testing method to check the ATV self-driving performance when following a paved path in the forest referencing an experienced human driver. The testing model adopts an Inverse Reinforcement Learning model that learns the best driving policy from an experienced driver in the environment of the paved path in the forest and outputs a reward function. The camera feeds the environment into a semantic segmentation model. The output reward function delivered a rating equation that can be applied to other driving techniques to measure the performance.

ACKNOWLEDGEMENTS

I would like to announce my gratefulness to God for his precious gifts.

I thank my parents and appreciate their sacrifices during the years to support me reaching this degree of achievements.

Also I would like to thank my advisor Dr. James M. Conrad, who guided me during my research and without whom this work would be impossible to achieve.

I am thankful for my committee members for their guidance and support: Dr. Aidan Browne, Dr. Dipankar Maity, and Dr. Chen Chen, I am also thankful for Dr. Hamed Tabkhi for his early guidance.

I acknowledge the help I got from all the graduate undergraduate students at UNC Charlotte who participated in this research under my supervision.

I would like to thank Michael Hanna at the NJ Institute of Technology and Shady Morgan for the mechanical engineering advice they provided me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ATV	All-Terrain Vehicle
AutoSAR	AUTomotive Open System ARchitecture
CAN	Controlled Area Network
ECE	Electrical and Computer Engineering
ECU	Electronic Control Unit
EPS	Electrical Power Steering
GPS	Global Positioning System
GPU	Graphical Processing Unit
H/W	Hardware
HWI	Hardware-Software Interface Layer
I2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
IRL	Inverse Reinforcement Learning
K15	Ignition Car Key
LIN	Local Interconnect Network
LR	Linear Regression
ML	Machine Learning
OEM	Original Equipment Manufacturers

PCB	Printed Circuit Board
PS	Power Steering
PWM	Pulse Width Modulation
RL	Reinforcement Learning
SAE	Society of Automotive Engineering
SPI	Serial Peripheral Interface
SS	Semantic Segmentation
UART	Universal Asynchronous Receiver-Transmitter
UNC Charlotte	University of North Carolina At Charlotte

CHAPTER 1: INTRODUCTION

The worldwide automotive industry is moving toward autonomous self-driven cars. The competition between inveterate mechanical car manufacturers and the new high technological electrical car manufacturers is increasing daily. Autonomous vehicles are still in an early phase of research, and those produced by some car manufacturers are expensive [1, 2].

Self-driven cars have numerous benefits and applications. These state-of-the-art machines save non-productive hours for drivers commuting daily to and from their work area to use them in more valuable tasks. They contribute to public transportation and shipments deliveries. More essential features are traffic safety and a massive reduction in the percentage of car accidents. All autonomous vehicles will be programmed to follow traffic laws. They will keep correct lane placement, respect the speed limits and the traffic lights. They will also be able to detect road obstacles [3].

Autonomous vehicles will lead to a significant reduction in the number of car accidents [4]. Another application uses similar machines to deliver emergency supplies in hurricanes areas, earthquake zones, and others. Those areas are not reachable by conventional vehicles, and the roads may be dangerous for human drivers. Another usage to be considered would be the discovery of new locations in the woods or deserts. While having autonomous off-road vehicles moving around, recording everything in their way, human operators can escape the danger of encountering wild animals, fallen trees, treacherous land features, or others.

This research focuses on one type of autonomous vehicle, making an All-Terrain-Vehicle (ATV) autonomous. The main goal is to make the ATV follow a paved path in a forest environment without any human interface. It is about using a standard

ATV and adds actuators and sensors to control the ATV and use Artificial Intelligence (AI) techniques.

1.1 My Contribution

The main contribution of this research is to provide an autonomous ATV that can follow a paved or gravel path using a camera and AI models and to provide a method to localize the ATV with respect to the path using a camera, semantic segmentation (SS), followed by an image processing model.

Another contribution is providing a new dataset gathered on a paved path in a forested environment from the walking trail at The University of North Carolina Charlotte that can help in more machine learning training and other type of research.

The following list illustrate my contributions:

- Created a new framework for training and implementation of an autonomous ATV.
- Used an existing dataset (DeepScene) [5] for SS which save an enormous amount of time and it was able to successfully represent the environment where the ATV was tested. The SS segmentation output was used into a novel image processing model to detect the ATV pose and orientation with respect to the paved path.
- Created a metric to quantify and compare the driving of an experienced to the driving of an autonomously controlled ATV.
- Implemented a system to take real-time images and control the steering of an autonomous ATV.
- Created a test bed for demonstrating these previous contributions.
- Provided a new dataset that contains 41059 entries. Each entry has a frame from the ATV camera showing the paved path, the current pose of the ATV at

this frame and the projection pose (both are calculated from the frame), and the corresponding steering angle of the ATV.

1.2 Previous ATV Work at UNC Charlotte

The University of North Carolina at Charlotte started working with Zapata Engineering in 2009 to make an Autonomous ATV. Zapata Engineering participated by giving the University a Honda TRX420FE 2009 ATV. Since then, researchers have placed a significant effort to make this ATV autonomous [2, 6].

In the previous research, the ATV had the actuators: throttle, braking, and steering controllers attached to the ATV. It also had to read from stationary sensors to identify the path. A hand-held remote control device commanded the vehicle over Controlled Area Network (CAN) messages [7, 8, 9].

1.3 Problem Statement

This research addresses transforming the existing system into a fully autonomous ATV without a human interface. Instead of controlling the ATV using a hand-held remote control device, this research sought to make it self-controlled using AI and Machine Learning (ML) techniques.

This research also addresses existing problems in the old system, like rebuilding damaged or missing boards, standardizing connections, communications, boards, power supply, the code used, and integrating the whole system.

In addition, this research addresses how to follow a paved path in a forest environment. It introduces the utilization of ML techniques and other AI methods as valuable methods of controlling the ATV steering angle and speed. It also addresses the problem of identifying a path in the woods and how to follow this path.

This dissertation also discovered the need to measure the driving performance on a forest path to evaluate the overall system for testing purposes. The research designed

and implemented a measuring system that was built using inverse reinforcement learning from an experienced human driver and measuring the performance accordingly.

1.4 Significance of this Research

This dissertation shall contribute to a proof of concept that it is possible to transform an ATV into an autonomous ATV with low-cost materials using AI modules.

As mentioned earlier, it is essential to have autonomous ATV for many reasons, including but not limited to the need to deliver emergency supplies to locations suffering from any catastrophic circumstances like hurricanes, earthquakes, or others that destroyed or blocked rural roads or when the roads are in an unsafe condition for human drivers to reach the final destination. Another reason is the need to discover non-rural areas like woods for civilization or search for lost campers.

1.5 Effort Toward My Contribution

The current research aims to build a fully automated ATV that follows a paved path in the forest without human interaction. This purpose requires the ATV to have a base architecture that digital signals from a processing unit can control. An embedded Graphical Processing Unit (GPU) is needed to process the data from a camera and take proper action in controlling the actuators installed on the ATV by sending the correct signals.

The research contains several steps: building the base ATV architecture, then building an AI model, which includes data gathering, choosing which models are needed, and training the models that need training. Then for implementation and testing, they are done first offline, followed later by real-time environment testing. Then, an AI model is created to quantify the ATV driving performance.

1.6 Effort Details

1.6.1 The ATV Architecture

The ATV architecture comprises a speed control system, a braking system, a steering module, a CAN network that connects everything, a camera, and an Nvidia Jetson Nano GPU. All modules have their processing unit and power supply. The effort includes enhancement and rebuilding of the existing braking system and throttle module, and redesigning and building a new steering module and a new CAN network.

- **CAN Bus and Power distribution:** Following the standard way of communication in the automotive industry, the different modules should be communicating using the CAN bus. The communication uses the CAN bus to connect all modules to the central processing unit. It is a perfect way to use the same bus to distribute power. This part of the research will be about physically designing and implementing the bus and the power distribution network while preventing the wrong connections.

- **Speed Control module:** The old throttle module controlled the ATV throttle by applying a fixed duty cycle to the servo motor attached to the air valve of the motor. This action gives constant power to the ATV to move forward. The problem with this design is that constant power does not mean a constant speed with the inclines in a forest. Therefore, the ATV needs the speed value as feedback to change the duty cycle accordingly. Several methods are available to get speed feedback. The first method used the encoders built before. Another approach pursued used GPS data. Finally, a successful method is getting the speed signal from the ATV primary shaft speed sensor. The signal is composed of a square wave, and the frequency is the indication of the speed.

- **Braking System:** The braking system is composed of a hydraulic actuator con-

nected to the braking pedal, a motor driver, and a microcontroller. The mechanical design is adjusted to have two pads sandwich surrounding the braking pedal where the hydraulic actuator is applied. This design is to protect the pedal from braking by distributing the weight over the whole pedal. The implementation of this system includes actuator calibration for the traveling distance.

- **Steering Module:** A problem appeared when using the old steering module. The old steering module uses the main power assist motor fabricated with the ATV. The power-assist motor works using a torque sensor that will define the engagement of the motor to help the human driver turn the handlebar. The motor is not powerful enough to turn the handlebar alone as Honda designed it only to help the driver turn. With the castle angle of the ATV, the ATV weight makes it very hard to turn the ATV right and left in a forward direction. To use this motor alone to steer the ATV, the ATV should be moving backward. As a solution, the research uses another controller with an external servo motor and an aluminum wire attached to the handlebar to steer it. The motor here acts like a human driver, and the power assist motor is just assisting in the usual way. The module gets a feedback signal from the handlebar axis to determine the current steering angle. The motor will adjust the direction accordingly to reach the proper required angle.

- **Base System integration:** A vital part of this research is integrating all previous modules. The CAN bus connects the GPU to the other modules. The GPU will handle the AI module and communicate with the steering, speed control, and braking system accordingly. The ATV should only run on the first gear with a maximum speed of 9 mph.

- **System Testing:** This implementation step includes three different types of testing with different strategies. Mainly, module testing is used for each module to

ensure it works alone without integrating it into the vehicle. Then, integration testing will occur by integrating two modules, then integrating three modules for testing until the integration is complete.

1.6.2 The AI Model and Image Processing System

The AI model and image processing system is reading a live stream from the camera and process the stream on a frame-by-frame basis. An SS model processes each frame to identify the path from the trees. The GPU processes the output segmented frames to determine the current ATV position and the projected ATV position on the paved path. An Linear Regression (LR) ML model uses the processed information and outputs the handlebar required angle to stay on the paved path [10].

- **Handling Camera Data:** An Intel deep scene camera D435i is mounted on the front of the ATV to gather the environment data needed. The research used Intel libraries and OpenCV to read the camera stream and transform it into the proper format needed.

- **SegNet SS model:** Nvidia Jetson Nano compatible SS model with CUDA enabled is used to determine the paved path in the forest. The model's training dataset is the DeepScene dataset which is a dataset for the forest environment. The CUDA enabled is a method to use the GPU's parallel cores in image processing to reduce total processing time.

- **Data gathering:** Using the camera and a sensor mounted on the handlebar axis, the ATV was driven on a forest paved path to collect data of the environment and the steering angle.

- **SS data processing:** The GPU processes the output of the SegNet SS model to get the ATV current and projected position. This dissertation defined an ATV status chart where the GPU transforms the SS output into numerical values.

- **Machine Learning Model:** A comparison between machine learning models. The LR model with Least Squares error calculation had the best results; therefore, this research uses it. The model's training uses 80% of the data gathered, and the testing uses 20% of the data.

- **AI Model Implementation and Testing:** Implementation of the AI was completed in two phases. The first phase was an offline phase where the camera captures the videos. The GPU applies the SS model to the videos. Then the GPU processes the output data from the SS model, and the software trains the ML using the training dataset and tests it using the validation dataset. The next phase was a Real-time implementation. In real-time, the research implemented a pipeline to treat frame-by-frame. This phase does not train the ML model, and it uses the ML model training results of the offline phase to determine the angle after each frame. The steering module receives this angle over the CAN bus.

1.6.3 Measuring Driving Performance

Measuring Driving Performance was the last step in this research. As a means of testing, the dissertation designed and implemented an AI model to measure the driving performance of the ATV. Its implementation uses the ENet SS model with IRL model. The main idea is to learn from an experienced driver and compare the driving technique to the experienced driver's technique [11].

- **Data gathering:** This research used a camera mounted on the ATV to gather data while an experienced driver was driving the ATV on a paved path in the woods. The research divided the data 80% for training and 20% for testing. The research gathered another dataset that does not represent good driving for testing purposes.

- **ENet SS:** While driving the ATV, the camera feed constructs a video to use later

to check the driving performance. The ENet SS model processes the video. The ENet is a SS model built on the Cityscape dataset. It is built in an efficient way to have a lite network with a relatively small size. The ENet is much slower than the SegNet with approximately 1 fps, but there is no problem with the slow performance because this one's usage is offline.

- **IRL:** The IRL is an AI model that knows the action policy and the environment. The IRL is trying to derive a reward function from the policy and the environment. This model's training uses the 80% set of the dataset of the experienced driver.

- **Rating Equation:** The reward function output of the IRL model concluded a rating equation. The software applies this equation to the SS output of the driving videos gathered and uses it to calculate the driving rating of the ATV.

1.7 Dissertation Limitations

This dissertation is limited to move the ATV following a paved path in a forest environment autonomously. The dissertation only assumes trees, green areas, and paths in the forest. This work assumes that are no obstacles on the paved path, like animals, humans, or large forest debris.

Also, this ATV under test is a manual gear shift ATV. This dissertation does not cater for automation of the manual gear transmission since the ATV will run on the first gear only with a maximum speed of 9 mph.

1.8 Organization

This dissertation contains seven chapters. Chapter Two provides information about the previous studies that already took place on this subject, the machine learning technology used in this dissertation, and the ATV initial status at the beginning of this research. Chapter Three describes the ATV architecture, the basic system on

which the AI control is controlling. Chapter Four describes the integration, testings, and results for the ATV base architecture. In contrast, Chapter Five illustrates the machine learning system design and performance goals. Chapter Six demonstrates the implementation and initial results of the AI system testing. Finally, Chapter Seven summarizes the conclusion, the lessons learned, and the future work.

CHAPTER 2: TECHNICAL BACKGROUND AND RELATED RESEARCH

This chapter provides information about previous studies on all the used AI techniques and related studies about controlling vehicles using ML and DL.

In order to understand the AI techniques used in this research, it is better to give a quick definition of the topics used, Machine Learning, Semantic Segmentation, and Inverse Reinforcement Learning, followed by a survey about the work related to measuring driving performance, then the work related to controlling ATV or standard vehicles using AI techniques.

Machine Learning is more concerned with learning from data to predict either a class or a specific value. Predicting a class is considered a classification, while predicting a value is categorized as a regression. In this research, ML is used to predict the rotation angle of the ATV's handlebar, which is considered a regression. ML models have two essential steps, training the model and using it. This research gathered a part of the data used for training the ML models used. The coming chapters explain the data-gathering process. After data gathering, data preprocessing may is essential to have the data ready for the training process. Examples of data preprocessing are handling missing data [10] or changing data format.

Semantic Segmentation (SS) is the process of linking each pixel to a labeled class [12]. It is a neural network that is composed of an encoder and a segmentation decoder. The training data is labeled data or labeled by color code. The SS model marks the output image with the required classes' colors. Fig. 2.1 illustrates an example for the SS output.

It is better to comprehend first Reinforcement Learning (RL) to understand the Inverse Reinforcement Learning (IRL). RL is a model with an agent in an environ-

Figure 2.1: Example for the Semantic Segmentation [12]

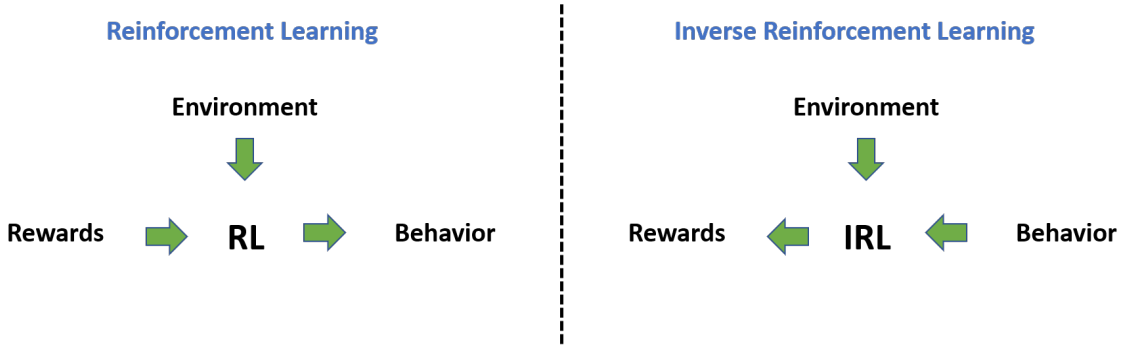

Figure 2.2: RL vs. IRL [15]

ment, a defined reward function, and an initial policy. The agent will act in the environment to maximize the reward function with different techniques to reach an optimum policy. In other words, the agent, the environment, and the reward function are available and trying to determine what is the optimum policy [13]. The IRL has the optimum policy as an input as well as the environment, and it aims to find the best reward function [14]. Fig. 2.2 illustrates the difference between RL and IRL.

Below is a survey of other related research.

2.1 Finding Reward Function for Reinforcement Learning

Reward functions describe the agent's best behavior. The reward functions explain what the agent should accomplish.

According to the actions taken, the environment moves from one state to another, determining the reward associated with the transition. The main goal of an RL agent is to collect the maximum amount of rewards possible. The agent can choose any action as a function of history or a randomized action.

2.1.1 RL with Unknown Reward Functions

The reward function is either precisely known or unknown. The authors of this paper [16] contributed by the derivations of the Bayes-optimal and mini-max policies in this setting and efficient algorithms for approximating these policies.

2.1.2 End-to-End Deep RL without Reward Engineering

Specifying a task to a robot for reinforcement learning requires substantial effort [17]. Most prior work that had applied deep reinforcement learning to real robots uses specialized sensors to obtain rewards or studies tasks that used the robot's internal sensors to measure reward.

2.1.3 Active Reward Learning

The authors of the paper [18] proposed learning the reward function through active learning, querying human experienced knowledge for a subset of the agent's rollouts. They introduced a framework wherein a traditional learning algorithm interplayed with the reward learning component. They demonstrated the results of this method on a robot grasping task and showed that the learned reward function generalized to a similar task.

2.1.4 Hybrid Reward Architecture for RL

This paper [19] contributed to a new method called Hybrid Reward Architecture (HRA). HRA took a decomposed reward function as input and learned a separate value function for each reward function, enabling more effective learning.

2.1.5 Faulty Reward Functions in the Wild

In this post [20], the authors explored one failure mode, which was when missing specifying the reward function. They highlighted what happened when a misspecified reward function encouraged an RL agent to subvert its environment by prioritizing the acquisition of reward signals above other measures of success.

2.2 Inverse Reinforcement Learning

Inverse reinforcement learning is a recently developed machine-learning framework that can solve the inverse problem of RL. IRL is about learning from humans. IRL is the field of learning an agent's objectives, values, or rewards by observing its behavior.

2.2.1 Algorithm for IRL

Russell & Ng. 2000 is a paper [14] on one of the early well-established linear IRL algorithms used to find the reward function. It had two different methods: one was a direct method for the small states space, and one used an approximation for the large states space. Their algorithm required as inputs the environment states, actions, and transitional probability.

2.2.2 Apprenticeship Learning via IRL

Abbeel & Ng 2004 [21] described an algorithm based on using IRL to try to recover the unknown reward function that terminated in a small number of iterations, and that even though they showed that they might never recover the expert's reward function. However, the policy output by the algorithm attained performance close to that of the expert, where here performance was measured referencing the expert's

unknown reward function.

2.2.3 Maximum Entropy IRL

Zibart et al. [22] developed a probabilistic approach based on the principle of maximum entropy. They provided a distribution over decision sequences to find a new approach to inferring destinations and routes based on partial trajectories. Mainly their research was more about finding a policy similar to the optimum policy of the expert behavior from the retrieved reward function.

2.2.4 Bayesian IRL

Ramachandran and Amir 2007 [23] showed how to combine prior knowledge and evidence from the expert's actions to derive a probability distribution over the space of reward functions. They focused on finding solutions for reward learning and apprenticeship learning tasks.

2.2.5 Relative Entropy IRL

In this paper [24], Boularias et al. studied imitation learning where the expert behavior covers only a tiny part of ample state space. They proposed a model-free IRL algorithm. This algorithm minimized the relative entropy between the empirical distribution of the state-action trajectories under a baseline policy and their distribution under the learned policy by stochastic gradient descent.

2.2.6 Nonlinear IRL with Gaussian Processes

Levine et al. 2011 [25] presented a probabilistic algorithm for nonlinear inverse reinforcement learning using Gaussian processes to learn the reward as a nonlinear function. Their probabilistic algorithm allowed complex behaviors to be captured from suboptimal stochastic demonstrations to get a simplified reward function.

2.3 Measuring Driving Performance using RL

2.3.1 Measuring Driving Performance by Car-Following

A new test [26] extended to measure impaired driving performance by such external factors as alcohol, medicinal drugs, mobile telephoning, and other sorts of distractions. Since attention and perception errors predominate over response errors in accident causation, on-road studies should specifically examine deterioration in attention and perception. Most existing methods of measuring impairing effects in the actual driving environment had the drawback that, irrespective of high sensitivity, they measured driving skills involved in only a low percentage of accident causes.

2.3.2 Standardized Definitions for Driving Performance

This research [27] developed names and definitions for 12 standard lateral and longitudinal driving performance measures relating to driving within and between lanes (lane departure, lane change, lateral lane position); steering wheel reversal; headway and gap (distance gap, time gap, distance headway, time headway); pedals (accelerator release time, accelerator to brake transition time, brake reaction time); and time to collision. The research developed human factors and lexicographical criteria for defining driving performance operational definitions. Based on the criteria and literature, the research developed common measure names and definitions. Human factors engineers and researchers would use these definitions for research and design, providing more consistent and comparable evaluation processes.

2.3.3 RL and Deep-IRL Planning for Autonomous Vehicles

This paper [28] focused on the planning problem of autonomous vehicles in traffic. The authors implemented a stochastic MDP representing the interaction between the autonomous vehicle and the environment. They also considered the driving style of an expert driver (i.e. experienced driver) as the target to be learned. The desired driving behavior of the autonomous vehicle is obtained by designing the reward function

and determining the optimal driving strategy for the autonomous vehicle using RL techniques. They collected several demonstrations from an expert driver to learn the optimal strategy using IRL. The research approximated the reward function of the expert using a deep neural network. It used the maximum entropy principle to learn the DNN reward function.

2.3.4 Driver Behavior Profiling

This paper [29] concentrated on driver behavior profiling. It aimed to find a low-cost to do that. It presented an investigation with different Android smartphone sensors and classification algorithms to assess which sensor/method assembly enabled classification with higher performance.

2.3.5 Risk Anticipation and Defensive Driving with IRL

This paper [30] focused on risk anticipation and defensive driving to ensure safety on residential roads. It provided a framework for modeling risk anticipation and defensive driving with IRL.

2.3.6 Predicting Driving Behavior using IRL

A part of this research [31] about measuring the driving performance presented an IRL framework with multiple reward functions to deal with environmental diversity. The model used Dirichlet process mixtures as a Bayesian model to divide the environment into clusters and simultaneously learned the parameters in each cluster.

2.4 Controlling Vehicles Using ML or AI

2.4.1 Automated Lane Detection by K-means Clustering

This paper [32] proposed an algorithm to detect lanes on roads using a camera in real-time and applying the K-means clustering method. The paper used the physical nature of the data to cluster the data. The paper interpolated the lanes to get the correct markings. The paper testings took place in different environments and

shadows [32].

2.4.2 Development of an Autonomous ATV for Surveillance

This paper [33] designed an autonomous ATV as a surveillance system remote controlled or autonomously controlled to follow a path at a port. The ATV used gears, a chain, and a motor to command the steering. A GPS and a compass planned the navigation. The ATV used a motor and a sensor to control the modified throttle. The ATV had audio-video feedback for the security system [33].

2.4.3 Fault Detection System for ATV

This paper [34] discussed the faults in diagnostics messages and the effect of the ATV electromagnetic noise on those messages, and how to get better, not faulty messages [34].

2.4.4 A Survey of DL Techniques for Autonomous Driving

This paper [35] surveyed the AI techniques used in autonomous driving cars and compared the use of deep learning (DL) models like convolutional neural network (CNN), recurrent neural network (RNN), and deep reinforcement learning (DRL). The paper inferred that the car needed to understand the environment through sensors like cameras and LiDar to have safe autonomous driving. As per this paper, the best way to understand was to use DL techniques to understand the camera/LiDar input and process it. DL techniques, including CNN, could help in building SS models like ENet and SegNet. According to this paper, the data gathered and used to train the model was the key to a good model. As there was no big problem in typical cases using traditional methods, the corner cases in driving were the ones that needed human intelligence. In this case, more training data were required [35].

2.4.5 DL for Obstacle Avoidance in Driverless Cars

This paper [36] used CNN with the internet of things (IoT) concepts to detect and avoid obstacles for autonomous cars. The paper attained 88% accuracy to detect and

avoid obstacles.

2.4.6 Road Tracking Using Deep RL for Self-Driving Cars

This research [37] used DL with RL, known as deep reinforcement learning (DRL) in autonomous vehicles, to track roads by self-driving cars. The researchers proposed a neural network to collect input states from the car's facing and produce suitable road tracking actions. This paper's approach achieved 93.94% driving accuracy.

2.4.7 Target Tracking of Self-Driving Cars

This paper [38] discussed the benefits of camera and radar sensor fusion in detecting missing objects in lousy weather. According to this paper, the fusion of the two sensors was more effective than using only one sensor in autonomous driving applications.

2.4.8 Object Recognition Technologies for Self-Driving Cars

In this paper [39], the researchers discussed sensor and object recognition technologies for self-driving cars and their requirements in terms of accuracy, unambiguousness, robustness, space demand, and of course, costs. The paper derived requirements on sensor technologies for self-driving cars.

2.4.9 ML based Traffic Light Detection for Autonomous Cars

This research [40] discussed different methods to detect traffic lights between different OEM's approaches of having a Google map with the traffic light locations or using cameras with Open CV2 techniques to detect it and to use IR as a backup method to detect the traffic light.

2.4.10 GPS-Based Steering Control for an Autonomous ATV

This paper [41] implemented an autonomous ATV with controllers for throttle, brakes, and steering that follow a GPS signal to follow a path in a field. The motivation of this research was to have a platform to test a crash protection device (CPD) to reduce injuries from ATV roll-over. In this research, the researchers used the electric

power steering motor to turn the handlebar. The main goal of the research was to allow the ATV to roll over many times to allow researchers to implement a better CPD [41].

2.4.11 Review on Navigation of Off-Road Autonomous Vehicles

The paper [42] categorized the navigation systems into six classes: image processing, neural networks, dead reckoning, statistical-based algorithms, fuzzy logic, and Kalman filters.

2.4.12 ATV Steering to Supplement Autonomous Functionality

In this project [43], the researchers proposed a modification to the mechanical implementation of the steering module of an ATV in order to be able to make it autonomous. Then the research considered the use of ML models as polynomial regression models to control the ATV steering angle.

CHAPTER 3: ATV ARCHITECTURE

This chapter describes the research effort to control the ATV using external signals. First, this chapter discusses what the previous researchers at the University of North Carolina at Charlotte implemented on the ATV. Then the modifications performed as a part of previous M.Sc. thesis and this dissertation work.

3.1 Previous ATV Work

The first paper published by the University of North Carolina at Charlotte in this project was published in 2010 [44]. The Zapata Engineering company initiated the research. They found the need for a Robot that can tow a trailer through non-urban roads. The Zapata Engineering company provided the University of North Carolina a Honda ATV for a senior design project in order to make it a remote-controlled device [6, 2].

In the previous research, a handheld remote control unit allowed the user to operate the ATV by sending commands to an evaluation board. This evaluation board controlled a throttle controller, a braking system, and a steering controller. The ATV also had to read from stationary sensors to identify the road [9, 44].

Many papers and research took place in the domain of controlling the ATV before. Below is a representation of some of the work done previously for different modules currently implemented and used in this research:

3.1.1 Throttle System

The actual throttle in the Honda ATV is designed as a wire-driven throttle with a spring return [44] where a wire is attached to a lever positioned at the right-hand thumb. This wire is connected on the other side by the air valve. When the ATV

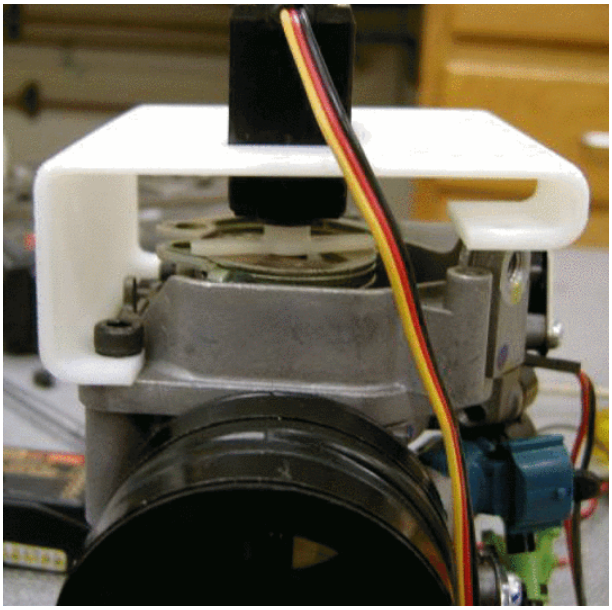

Figure 3.1: Servo attached to the Throttle assembly [44].

operator presses the lever, the air valve opens, allowing air and gas to go to the engine. The more pressure on the lever, the higher speed the ATV will acquire. In the first paper addressing this project, published in 2010, a parallax servo motor controlled the throttle. A bracket attached the servo motor to the factory original throttle assembly [44] as shown in image Figure 3.1. This paper controlled the ATV using a remote control that communicates with a Renesas evaluation board. The board was directly controlling the throttle module.

The change to the throttle model in this dissertation was changing the microcontroller used. At the start of this research, the throttle parallax servo motor was connected to the throttle original assemble as shown in Figure 3.1. It did not have a controller. This dissertation introduced a feedback control system to support automatic control. In the previous research, when a remote control commanded the ATV, the Pulse Width Modulation (PWM) duty cycle (DC) was enough to control the speed as a human was in control of the ATV's remote control.

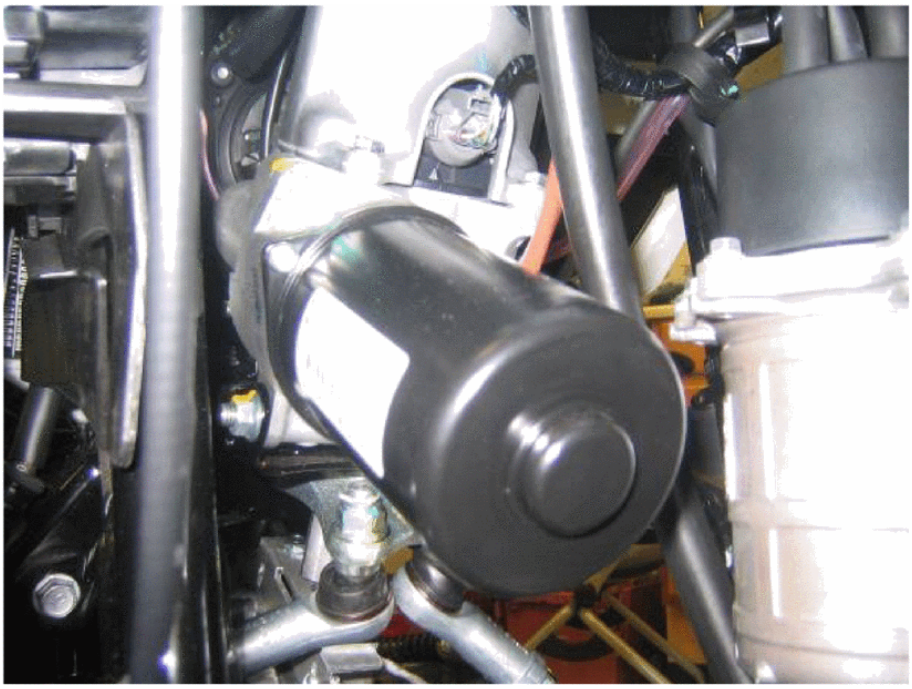

Figure 3.2: DC power steering motor [44].

3.1.2 Steering System

The Honda ATV has a Steering Assist module including a Torque sensor. This sensor is responsible for making the Steering Assist module start engaging to assist the steering or not. Figure 3.2 shows the motor implemented by Honda for the steering assist module.

The first paper published 2010 [44] assumed it could function the system by simulating the signal values to make it take control. This first paper had found it challenging to establish because of electrical issues with accurate values provided, and for the steering assistant, it depends not only on the torque sensor but also on the vehicle speed and other factors. This paper used a dual H-bridge as a motor controller for the steering assist motor. Figure 3.3 shows the circuit implemented in this research.

Another paper for the same ATV was published two years later [45]. This paper addressed a problem in the old design. The main issue for the first design was that the steering motor kept overheating. Overheating prevented the steering module from

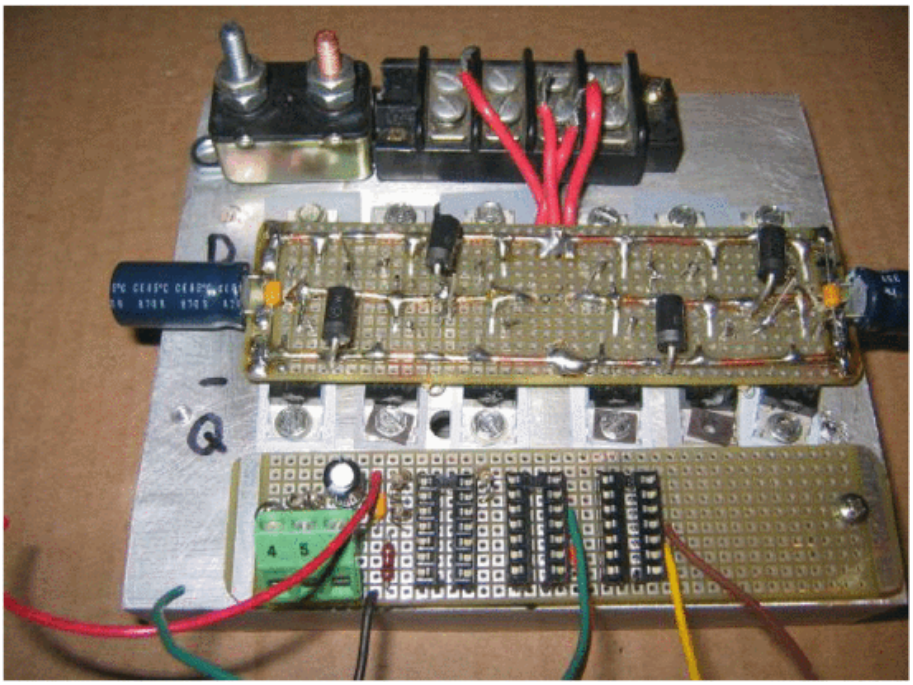

Figure 3.3: H-Bridge circuit attached to large heat sink [44].

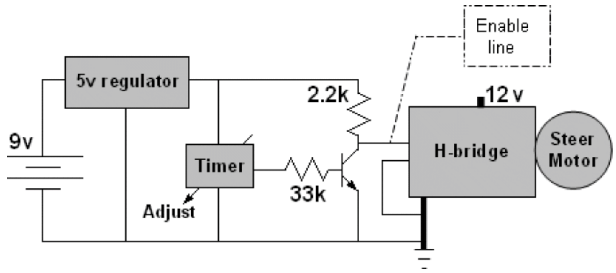

Figure 3.4: Steering PWM test circuit [45].

responding to all commands, and controlling the ATV was not perfect. It also may cause damage to initial ATV circuits. The second paper describes how the researchers implemented a PWM signal as input to an H-bridge instead of a continuous power signal. The PWM input signal reduces the amount of power input by the motor and reduces the overheating. A microcontroller introduces the PWM instead of having a constant voltage by a circuit. Figure 3.4 presents a diagram for testing the system using a PWM signal. Also, Figure 3.5 demonstrates the circuit diagram using the micro-controller.

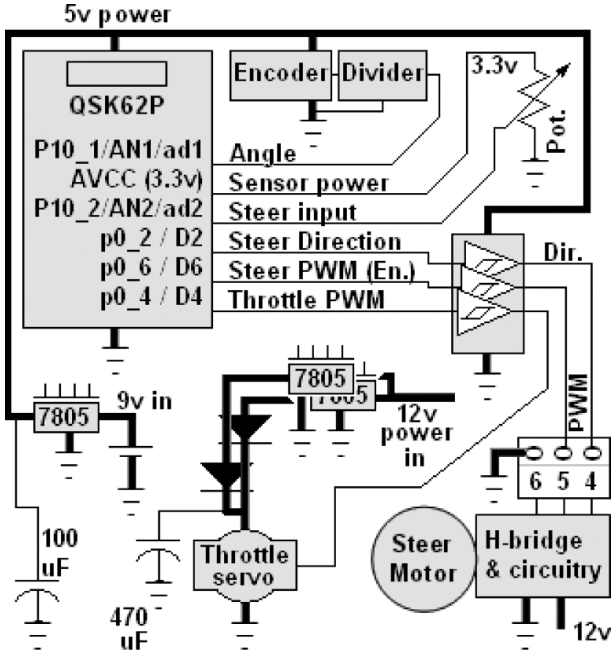

Figure 3.5: Steering circuit using micro-controller to control the motor using PWM [45].

The steering assist module of the ATV kept overheating since its design does not allow energizing the motor continuously. Honda planned this motor to work only in need according to the torque of resistance. A newer paper focused on the steering module [46] in which the torque sensor controlled the power assist module using the Honda's original controller. This design avoided overheating the motor by letting Honda's steering motor controller handle the motor as intended.

According to this paper [46], when the steering angle changes, detected on the operator bar, a resistance change appears on the sensor's output. The sensor has three output wires representing two separate output resistors. One resistor is for the clockwise angle value and another for the left side. The paper describes the usage of resistors and switches to simulate the sensor output to implement this controller. The researchers connected the neutral position resistance value replacing the actual sensor, then used the switches to introduce parallel resistors to change the resistance values.

This system successfully turned the steering to the right and the left. The implementation used analog switches, resistors, and a PIC micro-controller. This circuit could simulate the angle using a PWM signal to control the rate of opening and closing the switch. Therefore, the parallel resistances appear with a particular value that corresponds to a specific angle. The PWM signal is configurable using CAN messages.

A problem occurred when using this module in the current research because of the caster angle of the ATV [2, 6]. To obtain better control of the steering angle with the power assist motor of the ATV, the ATV should be going backward to use the instability of the wheel position as a helping force to control the handlebar. The current research introduces another solution to control the steering angle while moving forward.

3.1.3 Braking System

Two different braking systems exist in the Honda ATV. Two hand levers brakes that control the front wheels and back wheels, and foot pedal brakes that controlled the rear brakes.

The 2010 paper [44] provided a picture of the linear actuator used for the braking system with a customized H-Bridge. Figure 3.6 shows the braking system implemented as the linear actuator attached to the foot pedal and controlled by the customized H-Bridge.

Another paper published in 2014 [46] describes improvement of the braking system in the ATV. This researchers implemented the braking system using the same linear actuator in which a Pololu motor controller replaced the customized H-bridge to overcome overheating problems in the old one. This paper controlled the brakes using a Sakura micro-controller.

A recent paper in 2018 [8] mentions a study about controlling the ATV using Robotic Operating System (ROS). Before this study, the braking system was only

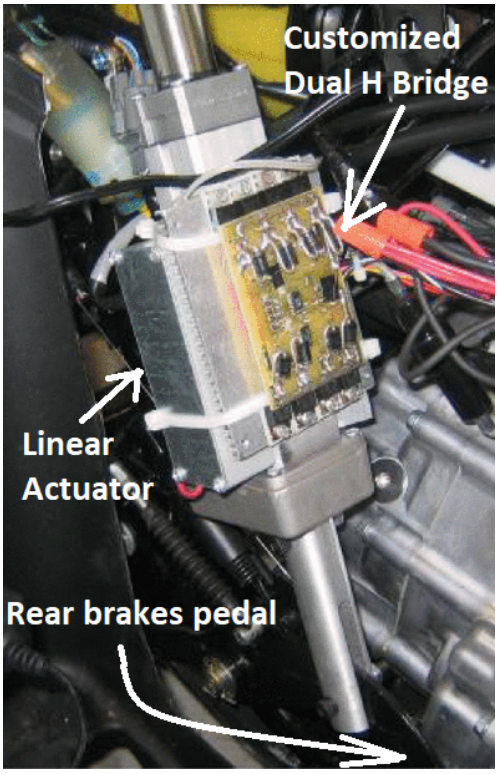

Figure 3.6: Old braking system applied in paper of 2010 [44].

either fully released or fully applied. In this research, using ROS, the braking system had three other stages, as shown in Figure 3.7.

The current research used the linear actuator and the aluminum support used to fix the linear actuator top end.

Time(sec)	Stroke Length (cm)	Brake Condition
1	4.1	Full Release
1.5	5.1	1/4 Brake
2	6.1	1/2 Brake
2.2	6.4	3/4 Brake
2.5	7.2	Full Brake

Figure 3.7: Braking system different stages table in previous research [8].

3.1.4 CAN Network

The Control Area Network (CAN) bus is a multi-master bus communication. CAN protocol is an asynchronous differential communication. It uses two signals for data: CAN-H and CAN-L for CAN High and CAN Low. The value transmitted is measured as the difference between the two signals. This communication protocol is advantageous in the case of the ATV. As the ATV is a machine that has some motors and an alternator to produce electricity, it is subject to electromagnetic interference. This interference could affect a signal on a wire which may give false messages. However, in differential communication, the interference will introduce noise to both wires in nearly the same way. While only caring about the difference between both wires, the difference value will not change as the noise adds a bias.

The automotive industry uses three major communication protocols: FlexRay, CAN, and LIN communication protocols [3]. FlexRay is the fastest, but it is costly compared to CAN and LIN. LIN is mainly used to communicate within one module when it is a complex module with many components. LIN is a one-master to many-slaves communication. The multi-masters communication employs the CAN bus because it provides a reasonable speed for normal - none timely critical - communications.

The CAN Bus consists of four wires, the CAN bus signals need two wires as mentioned above, and the power uses two wires. The first and last node should include 120 Ohm resistors between the CAN-H and CAN-L wires. If these resistors do not exist, the CAN messages will fail to propagate. The CAN bus determines the end of the bus by the resistance between CAN-H and CAN-L. If the resistance exists in the middle of the bus, the rest will not receive proper messages.

The CAN frame consists of a start bit, control signals, destination address, data, and a stop bit. There are two types of addresses: standard ID, an address of 11 bits, and extended ID, which is an additional 18 bits to be added to the standard ID to

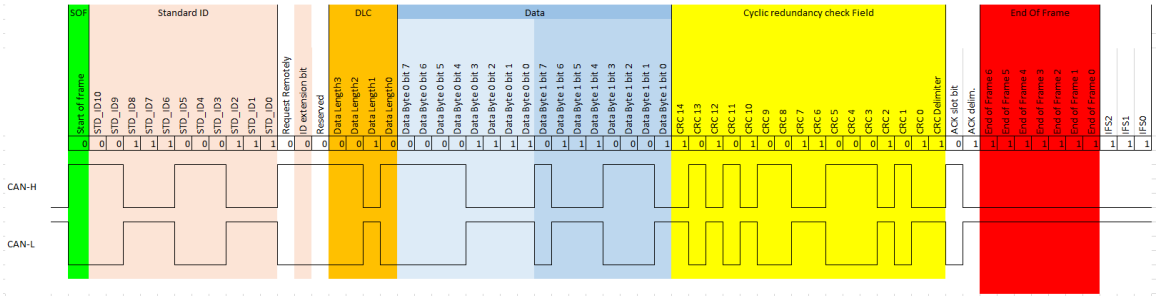

Figure 3.8: Sample of a Standard ID CAN Frame with 2 bytes of data

have an address of 29 bits total. Using only standard or both should be specified in the control signals. The data size in each frame can vary from 1 byte up to 8 bytes. The data size should be specified in the control signals as well. Figure 3.8 shows a sample of a CAN frame using Standard ID and sending 2 bytes of data.

The software must configure The CAN network to the same baud rate and sampling point. The baud rate and sampling point are configurable; however, it is more common in the automotive industry to use 250 kbps as baud rate and sample at 87.5%.

In the previous research, a paper written 2014 [46] was about using a CAN bus to control the ATV. This paper used the SAE J1939 Standard for CAN communication. The central processing unit communicates with the steering module using the CAN bus with an extended ID of 29 bits. The link used 250 kbps as baud rate and 87.5% as the sample point. This paper implemented a Sakura micro-controller with Texas Instrument SN65HVD251 Industrial CAN Transceiver for communication between all nodes with the central processing unit as Renesas RX63N. However, this implementation could not use the CAN APIs due to compatibility issues. The paper ended by using the CAN communication only between the steering module and the Central Control Unit board.

3.1.5 The Stationary On-road Sensors

The ATV in the previous research used to communicate with on-road sensors used as a SLAM solution for locations without Global Positioning System (GPS) signals. The on-road sensors draw the vehicle's path, and the human operator controlled the vehicle using a remote control device. The current research is not integrating those sensors [8, 9]. More recent research implemented stationary sensors for localization known as bread crumbs. This research planned for the future deployment of a receiver on top of the ATV. The receiver should read the ATV location from previously installed sensors in the ATV intended path when there is no GPS coverage [47].

3.1.6 The Speed Module - Wheel Encoders

As a way to communicate the ATV speed to be used with the steering module, a set of 2 wheel encoders was implemented and connected to the two rear tires. They were used to count the number of holes in the piece of plastic connected to the tire and count holes. It detects the number of turns, which helps to calculate the rotational speed. Knowing the tires dimensions and the rotation speed, it is simple to calculate the ATV speed.

3.1.7 Central Control Unit

The Renesas board RX63N was previously used as a central control unit. A remote-control receiver connected to the RX63N acquires the orders to control the throttle module using the three wires of the parallax servo motor. The central control unit also controlled the braking system by providing a signal for the H-bridge for the linear actuator. Besides, the Renesas board was controlling the steering module by sending the CAN messages to the steering module PCB [7, 8].

There was no physical connection between sensors and actuators at this point except for the steering module that was getting readings from speed sensors. Also, this central control unit does not make its own decisions to control the ATV as it

was not processing the sensors data. Instead, it was getting the commands from the remote control used by a human operator who processed the sensors data and acted accordingly.

3.2 ATV Current Architecture

As previously mentioned, this research consists of two significant steps. The first step is implementing a base ATV architecture that controls the basic driving functionalities using digital signals. The second step is to implement an AI model to control the ATV.

This section discusses the ATV architecture and the control of the fundamental driving functionalities like the speed control illustrated in the throttle and the braking system and steering control. It also discusses different ideas of implementations with a comparison between them. This section also demonstrates the whole ATV CAN network implemented.

As the main idea for this section is to mimic a human driver's action. After observing the human driving behavior, it is evident that the human operator controls the ATV using a throttle lever, braking levers and a breaking pedal, and the handlebar. The operator controls the speed by changing the throttle lever pressure, and in some cases, the operator might need to press the brakes. The operator releases the throttle's lever and uses the brakes to stop the ATV. To change the ATV direction, the operator turns the handlebar to the direction needed. The ATV driver uses their eyes to gather data about the environment and then decide the proper action.

According to the observation and the target of this chapter, actuators are implemented and installed to control the ATV in the same way a human operator will be operating it. This dissertation implemented a throttle controller to control the speed, a steering controller to control the steering angle of the ATV, and a braking system to control the brakes. Similar to the human eyes, the ATV uses a camera to gather data about the environment. The GPU processes the data gathered and makes

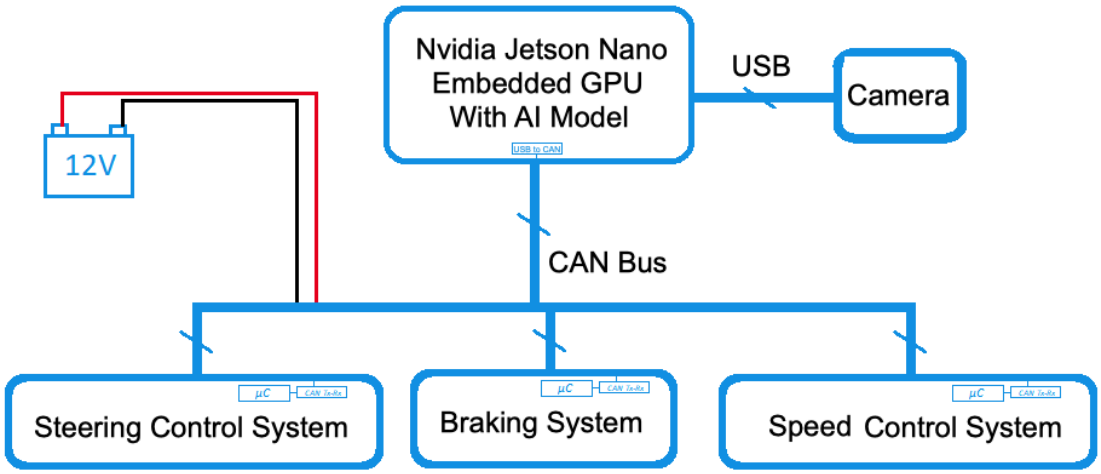

Figure 3.9: System overview

proper decisions. The next chapter discusses decision-making. As in the observation, only one operator is dealing with the ATV. Therefore, a CAN bus connects and synchronizes all actuators, sensors, and the GPU. This dissertation chose the CAN bus as it is a multi-master bus with a lower cost than the FlexRay. Its average speed is currently suitable to the speed required for the planned system with the expected data rates needed except for the camera connected directly to the GPU using a USB 3. Figure 3.9 illustrates the system overview for the current research.

As shown in Figure 3.9, the system consists of multiple nodes working with the idea of decentralized processing units to reduce the processing power needed by the central/graphical processing unit and reduce delays. A CAN bus connects the whole system. It is responsible for data transmission and power distribution. The system consists of actuators represented here as brakes, throttle, steering controllers, and sensors represented by the camera. An Nvidia Jetson Nano is responsible for synchronizing the system by reading the data from the camera and reacting by sending orders to controllers through the CAN bus. Controllers are only receiving the CAN messages as orders, and each controller is responsible for order execution. The sys-

tem is implemented in phases, implementing standardization library first for MSP430, then implementing the CAN module and testing it, followed by the implementation of actuators in a standardized enhanced way [2, 6] illustrated in this dissertation, after that adding the sensors one at a time.

As part of standardization, the design shown in Figure 3.9 includes in each node an MSP430 microcontroller. An intermediate PCB connects the MSP430 to a CAN shield with an MCP2515 CAN transceiver. This PCB is acting like a motherboard with a power supply.

The sections below explain in detail each module design choice and implementation.

3.2.1 CAN Module

The CAN module implementation consists of: the physical bus wires, the CAN shield connection to the MSP430, and the software configuration of the CAN Module. Below is an illustration of the different parts of the CAN module implementation:

3.2.1.1 Physical Design and Implementation

The system designed has the primary processing unit, connected to separate small processing units that control and handle different nodes. A CAN bus connects these nodes and the primary processing unit as illustrated in Figure 3.9.

For this project which has been running for twelve years, to prevent effort loss and repeating work, this implementation has Poka-Yoke CAN plugs that can be plugged-in in only one way to prevent faulty connections. Figure 3.10 demonstrates the plugs.

Each plug has four twisted wires [48] to reduce interference: CAN High (Yellow), CAN Low (Blue), High Voltage of 12 Volt (Red), and Ground (Black) as shown in Figure 3.11.

3.2.1.2 Hardware Configurations

The hardware configuration uses a CAN shield with an MCP2515 CAN transceiver chip that communicates via SPI with the MSP430F5529 microcontroller. A PCB acts

Figure 3.10: CAN Bus physical implementation: CAN connectors

as a motherboard to connect the two boards and be used as a power supply to convert the 12V ATV battery into 5V to supply the microcontroller, and the CAN shield.

3.2.1.3 Software Configurations

The software used SPI to write the configurations for the CAN Shield [46, 49, 50, 51, 52]. The software allows different configurations such as Standard ID or Extended ID frames. The first configuration and testing used Standard ID frames. However, when started to use the old steering module, it is configured to Extended ID [46].

The CAN transceiver configurations referred to the MCP2515 datasheet and to some online examples that were modified to match this project [53, 54, 55].

To test the CAN module after the configuration, an MSP430 board sent 0xAA to another, and the second board turned a LED on when it received 0xAA, which means it works fine. Below is an image from the oscilloscope showing the full-frame used to test the board. The image was using a standard address of 0x0181. The CAN module has many factors that contribute to the proper functionality, and any of them may

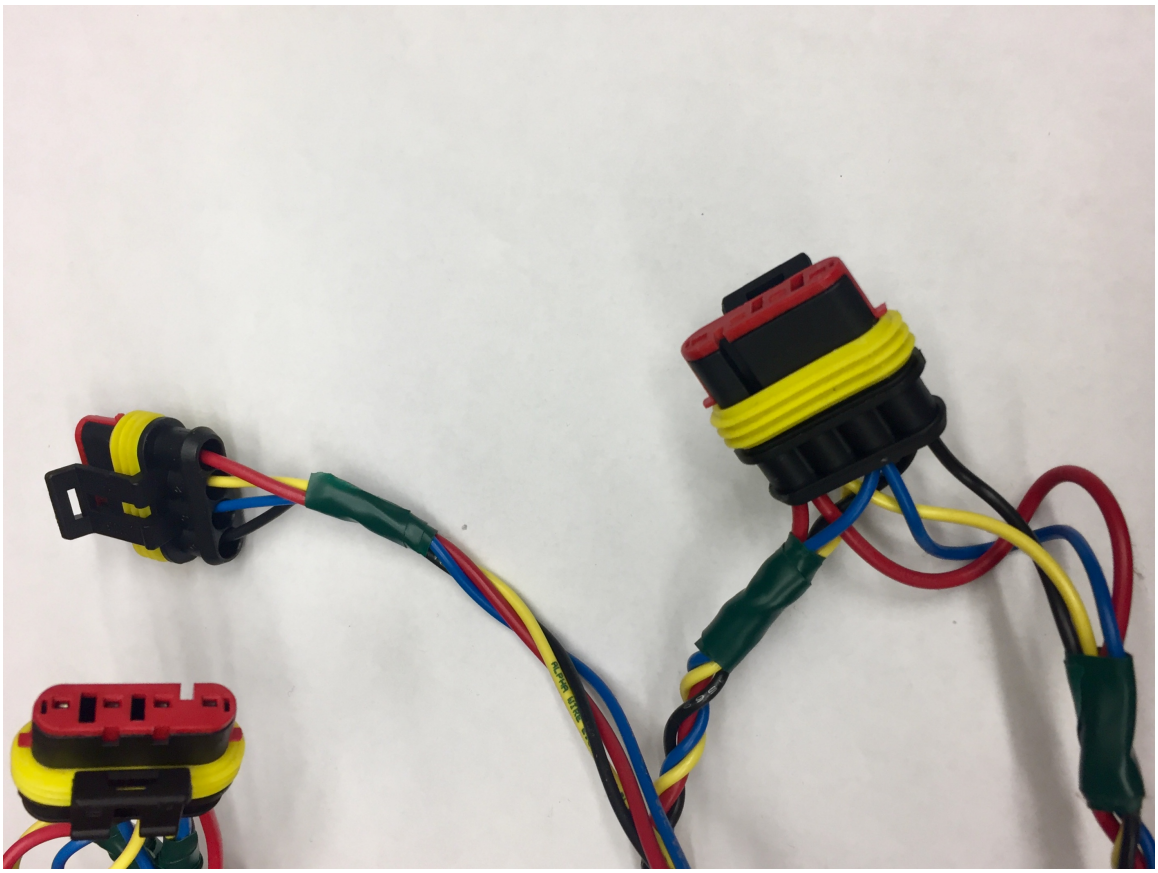

Figure 3.11: CAN Bus physical implementation: CAN Bus after assembly

lead to system failure. It is always preferred to configure and test only one factor at a time.

3.2.2 The Braking System

The ATV used is a rear-wheel-drive vehicle, meaning the engine controls the rear wheels. The research chose the rear braking system controlled by the foot pedal as the braking control system. Therefore, stopping the rear wheels will help stop the whole ATV. Furthermore, it is safer to stop the rear wheels first to prevent the ATV from turning over. If the automated braking system were using the front brakes, the front wheels would stop while the back wheels remain running with the engine power, possibly causing the ATV to roll over toward the front direction.

A 12V linear actuator controls the foot brake. It is capable of 100 lbf with a 4-inch

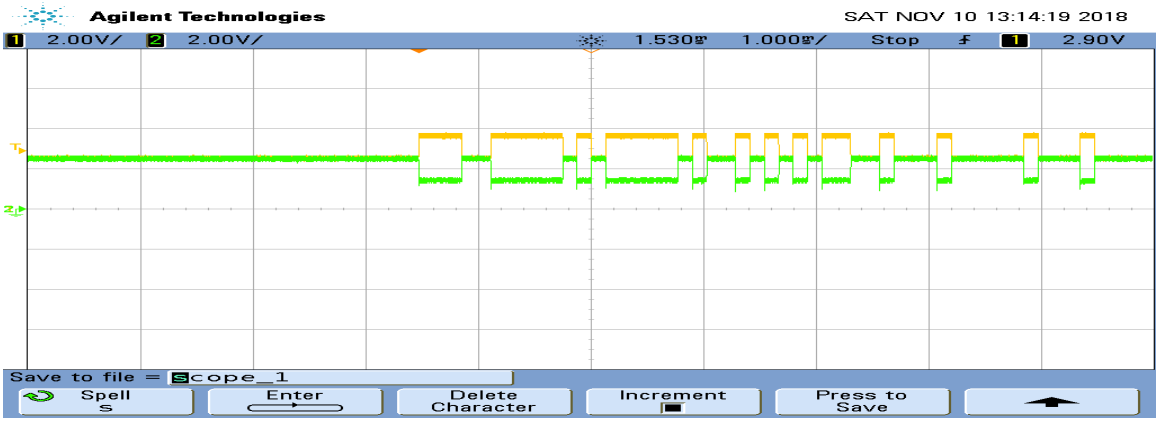

Figure 3.12: CAN frame on oscilloscope, the yellow wire is the CAN-H, and the green wire is the CAN-L

travel range. The actuator is connected to a rigid mounting bracket affixed to the chassis tube frame and a custom adapter bracket that mounts to the existing foot brake pedal [6].

3.2.2.1 Actuator

The actuator is equipped with limit switches to restrict extension and retraction to an appropriate range for the brake pedal. The brake pedal's position with no load applied and under full braking by a human operator determined the placement of these switches.

An off-the-shelf dual H bridge motor controller commands the actuator. It is rated for a maximum voltage of 30V and with average current outputs of 18A and stall currents of 40A.

3.2.2.2 Actuator Extension

Quick-release pins pinned the actuator extension in place at the actuator and the bottom mount bracket. Removing these pins allows a test operator to quickly disengage the system and regain manual control over the brake system, as needed [6]. Figure 3.13 demonstrates the quick release pins holding the linear actuator in the two pieces bracket sandwich implemented around the brake pedal. Figure 3.14 shows the

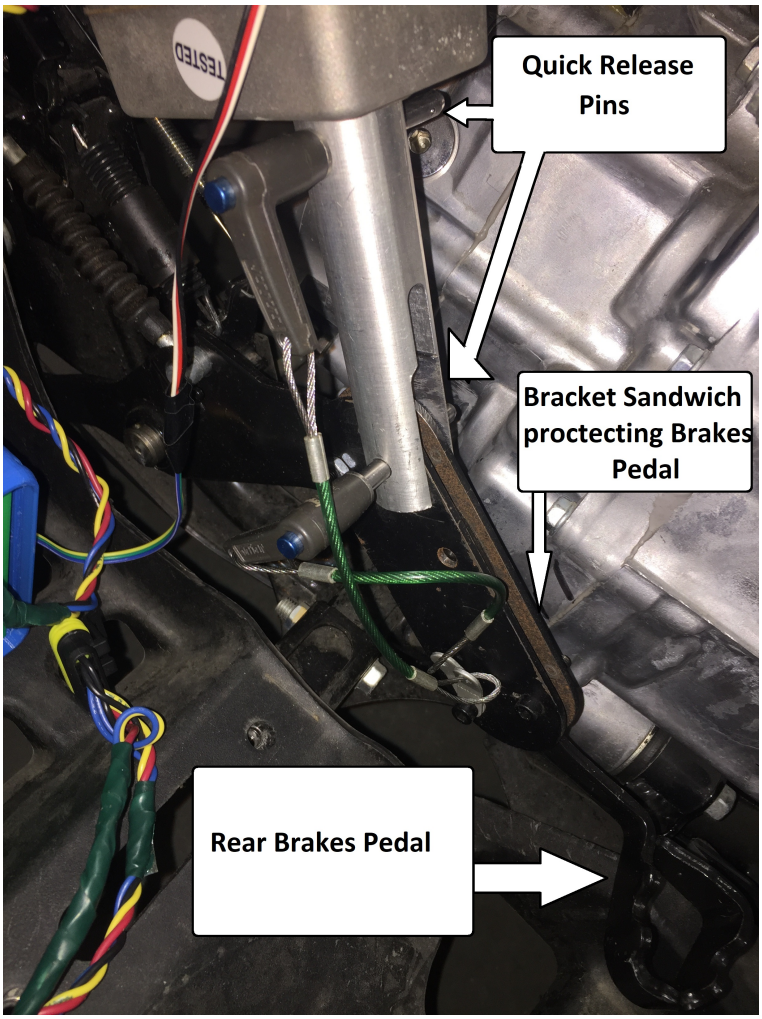

Figure 3.13: Brake Pins fixing the linear actuator to the two-piece bracket sandwiching the brake pedal

quick release pins released, and the braking system unmounted.

3.2.2.3 Bottom Mount Bracket

The bottom mount bracket ensures the force applied by the actuator is adequately constrained, only influencing the rotation of the lever about its mounting axis and not allowed to excerpt force in any out-of-plane direction. The stock brake lever is a "free form" sheet metal part without constant geometry that would be suitable for quickly mounting brackets. To solve this issue, a two-piece bracket that sandwiched the lever using three bolts positioned in direct contact with the top and bottom of

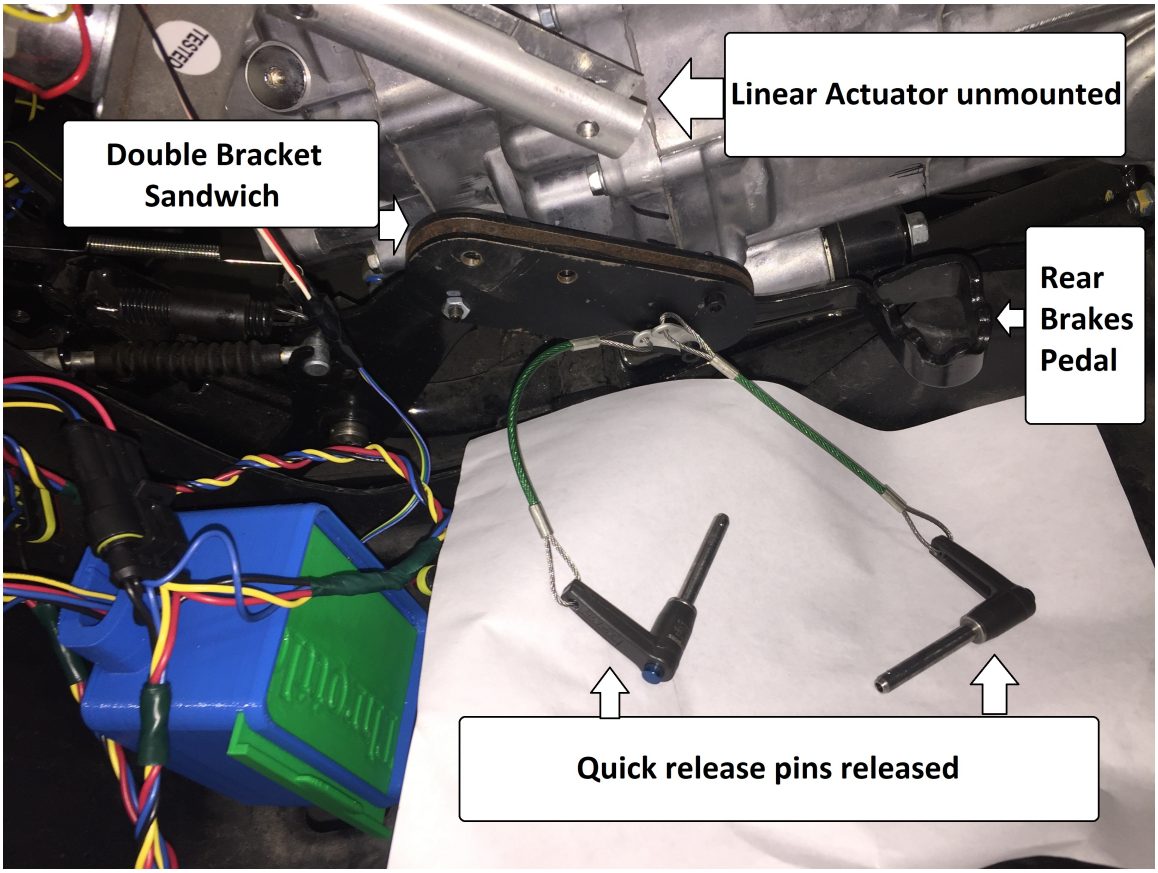

Figure 3.14: Quick release pins released and the braking system unmounted

the lever evenly distribute the force applied to the bracket through the quick release pin linkage [6] as shown in Figure 3.13.

3.2.2.4 Software Design and Implementation

An MSP430 controls the braking system and uses a CAN Shield to connect it to the ATV network. The MSP430 uses I^2C communication to control the dual H bridge. The MSP430 transmits signals to apply and release the brakes to the motor driver, who sends PWM signals to the linear actuator. While the actuator has one start point and one-stop point, the PWM involves the stop switch as it gives the motor a sudden start with each new cycle of the PWM. The signals set the timing to apply the brakes to 1.1 seconds and release the brakes to 1.7 seconds to prevent damage.

Consequently, applying the brakes for the same amount of time to release the brakes

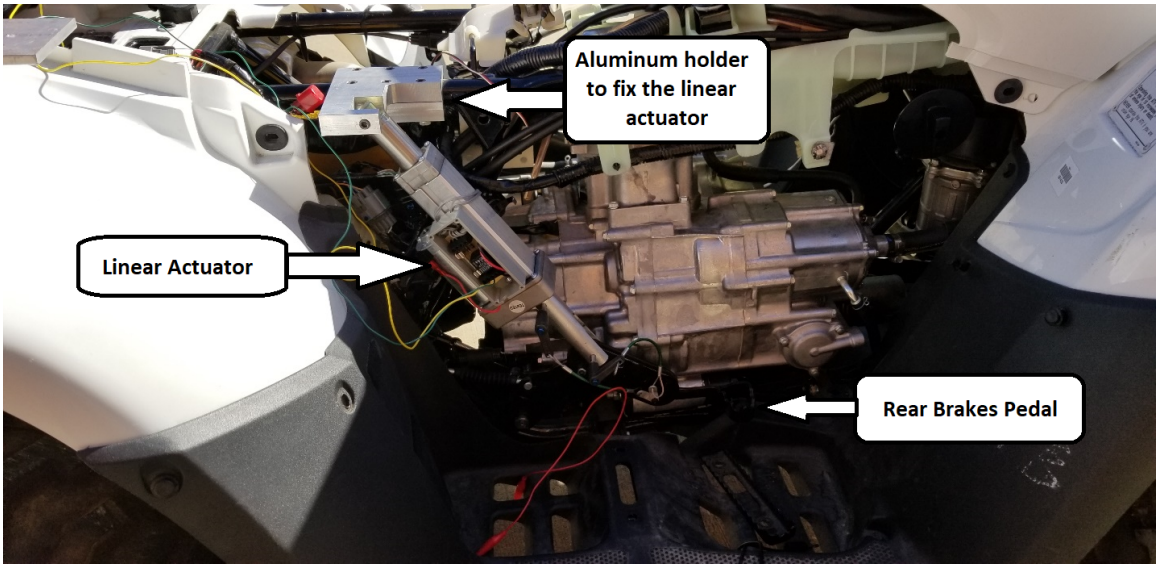

Figure 3.15: Braking system installed on the ATV

causes the start point to move slightly every time the braking system operates. That is why the time the release signal is applied is more extended than when applying the brakes. This timing difference ensures that the brakes will return to the same initial point. After sending the release or the apply signal, the controller sends a signal to stop the motor movement. The central processing unit sends the signals to the MSP430 of the braking system.

The braking system sets a flag for the last status. If the status of the braking system is the same as the command received by the central processing unit, the braking system will not act. If the braking system status is different, it will apply the new command and change the current status. For example, if the brakes are already pressed, and the central processing unit sends a CAN message to the braking system to press the brakes, the braking system will not perform any action. If the brakes are pressed, and the Central Processing Unit sends a CAN message to release the brakes, the braking system will send an I^2C message to the dual H-bridge to release the brakes for 1.7 seconds, then it will send another I2C message to stop the brakes and will change the status flag to be released.

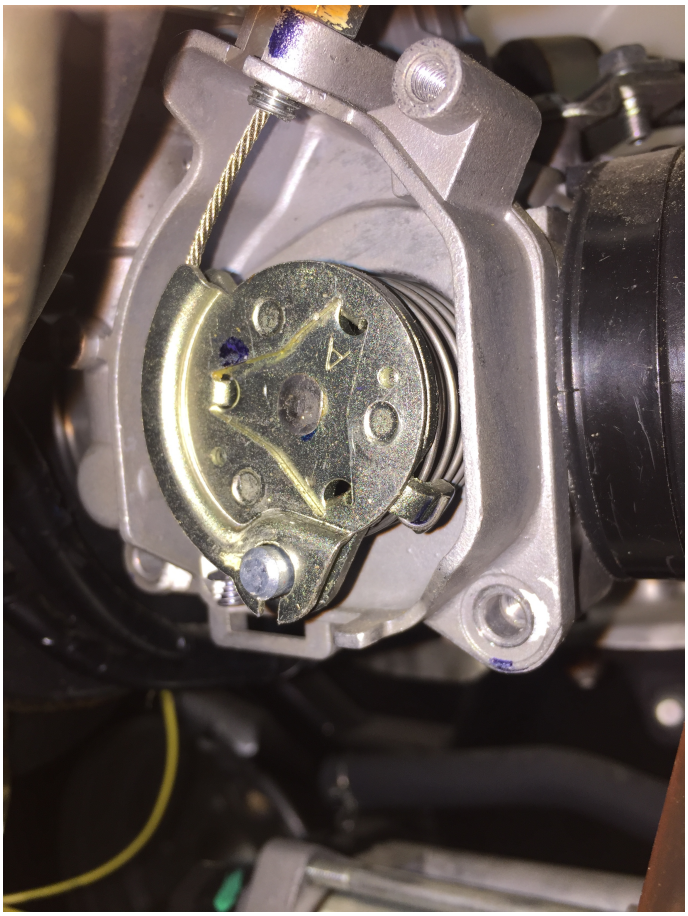

Figure 3.16: Throttle wire and spring at the air valve side

3.2.3 Speed Control System

Honda manufactured the throttle in the ATV as a lever to be pressed by the right-hand thumb. A wire connects it to the air valve that, when opened, leads to open the gas valve and allows the gas pump to provide more fuel according to the angle the lever moved. The more pressure the button, the more fuel is injected, the higher speed the ATV should go. If the lever is free from pressure, it is equipped with a metal spring to bounce back to the initial position. Figure 3.16 illustrates the original part equipped by the ATV, Figure 3.17 shows the servo motor with the plastic casing to mount it in the ATV. Figure 3.18 demonstrates the servo mounted on the ATV.

As the implementation is purely mechanical, the former team installed an electrical servo motor at the air pump terminal of the wire as shown in Figure 3.17 and

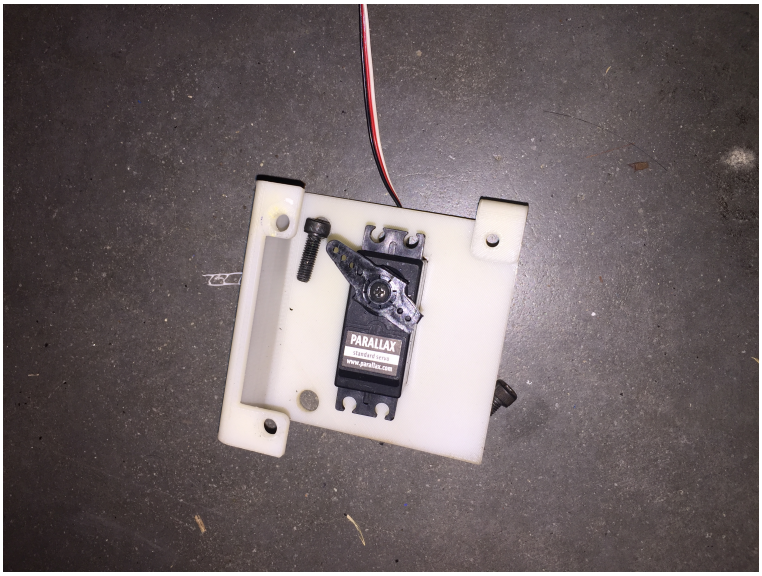

Figure 3.17: Servo motor used to control the throttle with a plastic casing to fix it around the original throttle parts.

Figure 3.18. In the current implementation, a PWM signal controls the motor. According to the datasheet, it can have a duty cycle from 3.8% up to 11% to move the motor 180 degrees. The original wheel shown in 3.16 can only move 90 degrees, which is duty cycle from 3.8% up to 7.4%. However, because this research is only using the first gear, and to avoid overloading the engine, a decision is made to limit the duty cycle to 6% maximum.

This implementation controls the throttle by changing the duty cycle value. After testing this implementation on a path in the forest, it is clear that the duty cycle only controls the engine power, not the ATV speed. According to the area topography, if the path is horizontal, there is no problem, but the speed will increase if it starts to go down. If the path goes up, while the duty cycle is not high enough, the ATV will stop moving as it is not getting enough power.

The conclusion was to change the model from a throttle control into a speed control system, a closed-loop control system with feedback from the ATV speed. In order to get the speed, the research considered the following approaches.

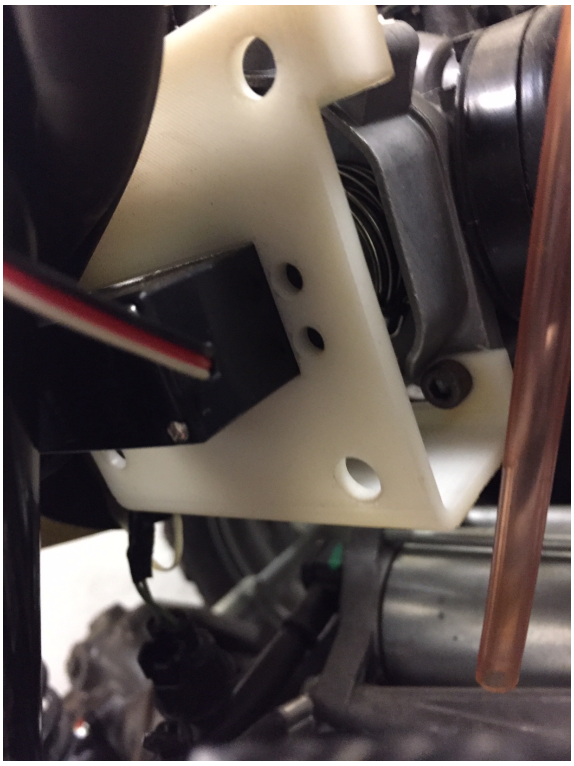

Figure 3.18: Servo motor attached to the air valve controller

3.2.3.1 Using a GPS Module to Get the Speed

This part of the research used an Adafruit GPS to measure the speed from the GPRMC messages from the GPS. Testing the GPS showed that the GPS provided the speed successfully. The research implemented a module to control the duty cycle for the servo motor using the speed from the GPS. This model was not that effective for three reasons:

- The time the GPS can fix on satellites is between 30 seconds and 30 minutes. This latency was not reliable.

- GPS update time was longer than required (5 seconds on average), which made the idea of feedback impractical. The throttle controller may increase the speed to the maximum allowed duty cycle value before getting feedback. A workaround is to reduce the acceleration and deceleration time to match the

time of GPS speed updates; however, this workaround is impractical.

- The main research is to make the ATV follow a paved path in the forest. It is not probable to fix GPS on satellites in a forest environment with all interleaving leaves and trees.

Therefore, the research searched for another solution to get the speed.

3.2.3.2 Using the ATV Engine Speed Sensor to Get Speed

The research decided to use the signal from the speed sensor attached to the engine to get the speed value, as shown in Figure 3.19. The sensor outputs signals in a square wave format where the signal's frequency represents the rotation speed of the motor shaft before the gearbox. In this case, counting the pulses for a period of 350 ms provided the speed. The closed-loop control system implementation using the speed sensor must respect timing to avoid overshooting and follow the command in the fastest stable time. In this case, the fastest time without overshooting is to read the speed every 350 ms and adjust the duty cycle with 0.1% every 20 ms, resulting in a step of 1.7% duty cycle change per reading and covering the maximum range in 0.7 seconds. While targeting a maximum speed of 6 mph (2.68 m/s) (at first) will ensure the required speed to be achieved in less than 2 meters of travel (1.87 meters); though, as shown later, the vehicle operated at a lower speed of 2.9 mph. This final implementation is the most reliable method that does not depend on external parameters; therefore, the dissertation uses it moving forward.

3.2.4 Steering Module

At the beginning of this research, this module was re-used from the old research with minor enhancements for the connector to make the whole system use the same CAN bus connections. The already-existing module implementation used a PIC microcontroller. The microcontroller receives messages using CAN transceiver MCP2551 to communicate the angle value to the PIC using SPI signals. The PIC controls an

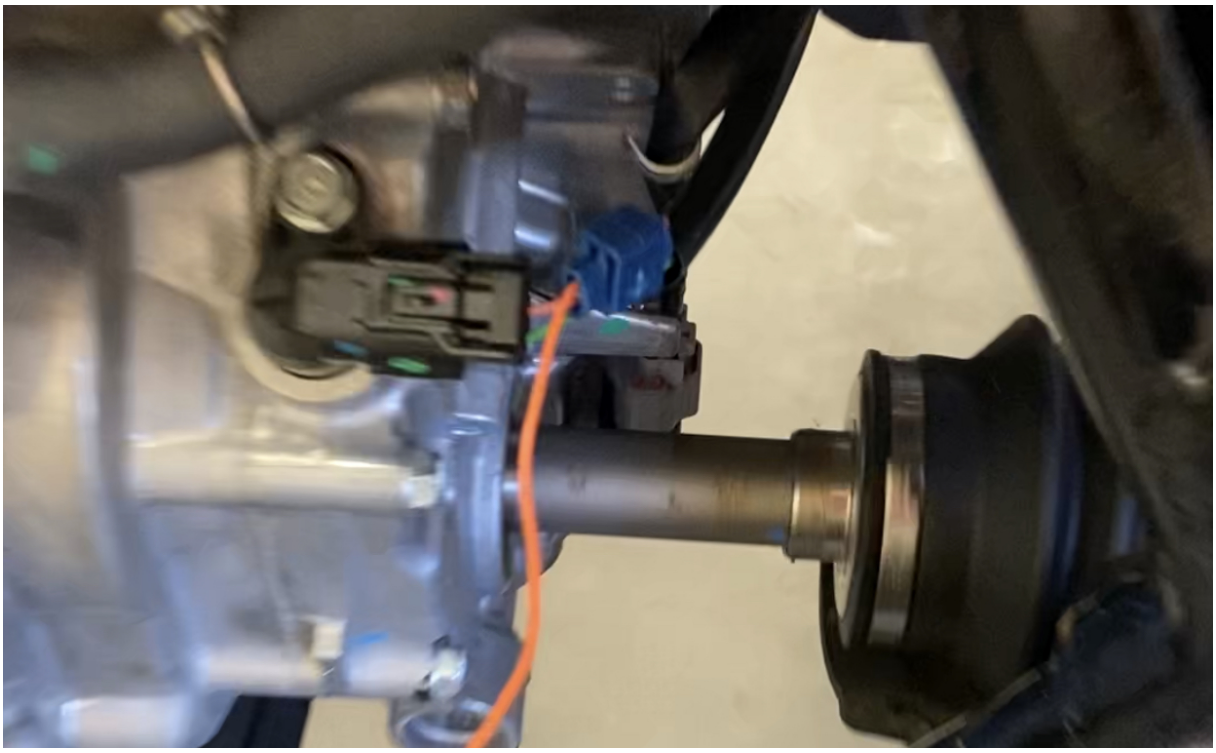

Figure 3.19: Speed sensor attached to the engine

analog switch that connects and disconnects some parallel resistors to simulate the Torque sensor values using a PWM signal [45, 46]. The main idea is to simulate the torque sensor output to trigger the system pretending to turn under high resistance, which will lead the steering assist motor to engage and turn the steering wheel. Figure 3.20 shows the old ECU used to control the steering, while Figure 3.21 demonstrates a CAN message used to control the steering.

After connecting the system to the vehicle, the controller over CAN bus can successfully move to the right and stop at a certain angle, move to the left and stop at a certain angle, and back to the center position and stop with the ATV suspended above the ground.

A simple test for the steering module is to make the ATV turn in one complete circle. Some calculations used the steering dynamics theories with the Ackerman steering condition (having a different angle for the inner wheel different than the outer wheel to all a turn with a slip-free) [56]. Figure 3.22 demonstrates a turning

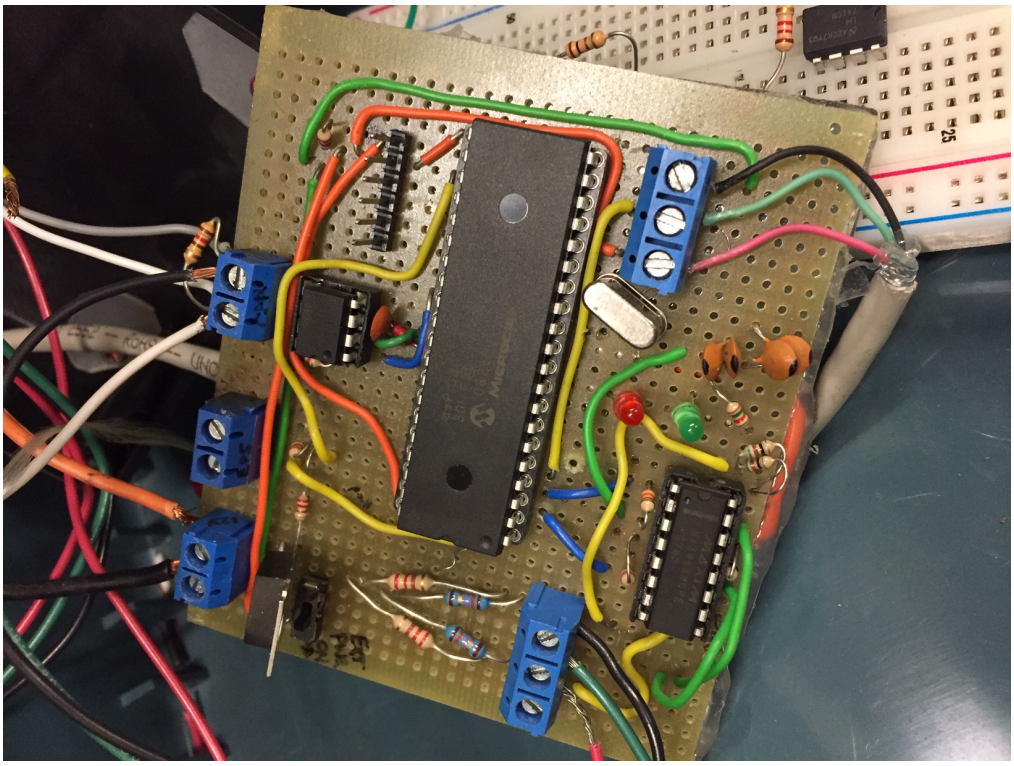

Figure 3.20: Steering module

vehicle. In Figure 3.22, L is the distance between the center of the front wheel and the back wheel, in this case, $L = 50"$. θ is also known to be $30°$ from the testing of the ATV while suspended above the ground. The following calculations derived the time to do one full circle:

$$
\begin{aligned}
b &= 90 - \theta \\
b &= 60^\circ \\
\cos b &= \frac{L}{H} \\
H &= \frac{L}{\cos b} \\
&= \frac{50''}{1/2} \\
&= 100'' = 2.54\text{m}
\end{aligned}
$$

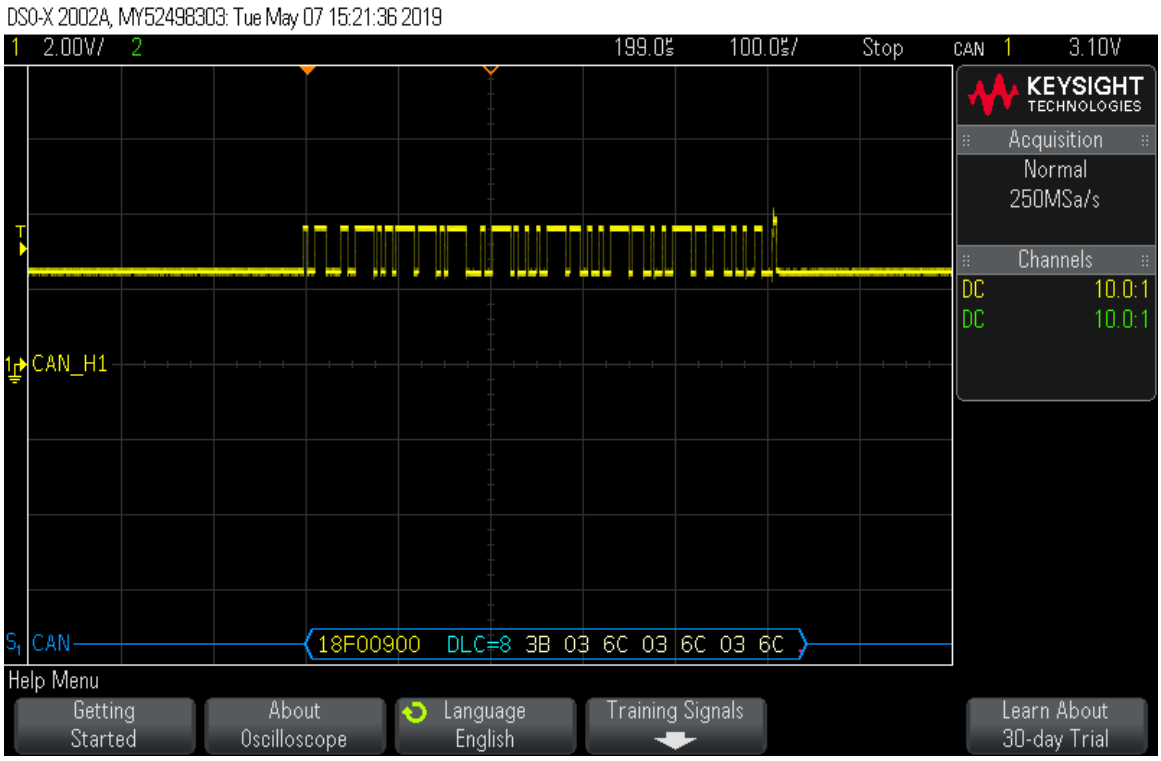

Figure 3.21: CAN message to turn steering right

$$
\begin{aligned}
CirclePerimeter &= 2\pi \times \mathrm{H} \\
&= 628.318'' = 15.96\mathrm{m} \\
Speed &= RPM \times TirePerimeter \\
&= \frac{80 \times 0.6 \times \pi}{60 \text{ seconds}} = 2.51\mathrm{m/sec} \\
v &= \frac{\Delta d}{\Delta t} \\
\Delta t &= \frac{\Delta d}{v} = \frac{15.96}{2.51} = 6.35\mathrm{sec}
\end{aligned}
$$

To turn in a circle, the ATV moves forward, turns for 6.4 seconds, then stops. The ATV front tires rotated only 10° instead of 30° when the central unit instructed the steering module to turn the maximum value to the right. The resulting circle was 15

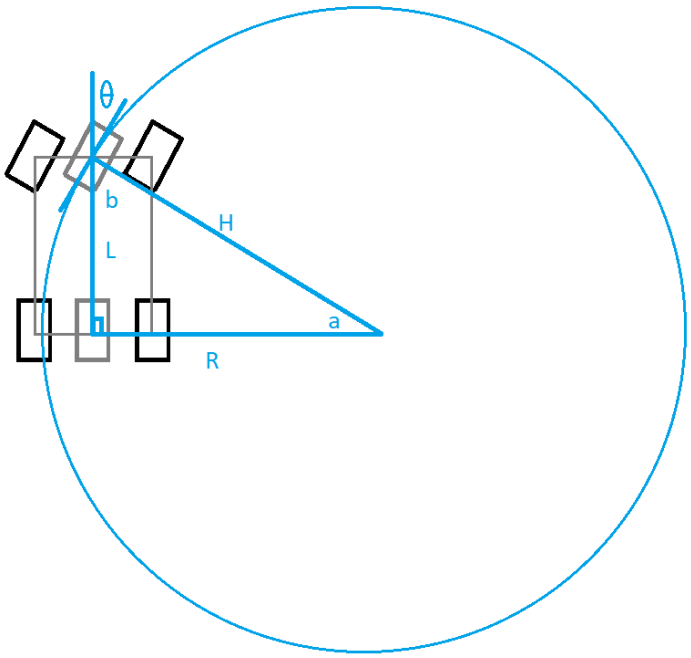

Figure 3.22: A turning vehicle

meters in radius, and it would take more time and distance to have the circle done.

The same test was done again while putting the ATV the reverse gear with the maximum right angle steering value. The ATV steering angle was 30°.

The researcher did further investigations to understand why the ATV has a larger turning angle while going backward than forward. In conclusion, there are two main reasons:

- The first one is that the ATV has a power assist module to help a human operator turn the steering wheel easier when the ground resistance is high. The EPS needs human operator applied torque to work. It can not move the steering wheel alone, so the motor power is not strong enough to make it work by itself.

- The second reason, which makes it turn easier when the ATV is going backward, is the Caster Angle [57]. The ATV front wheels axes are not perpendicular to the ground. The axis has an angle called Caster Angel. This angle's purpose

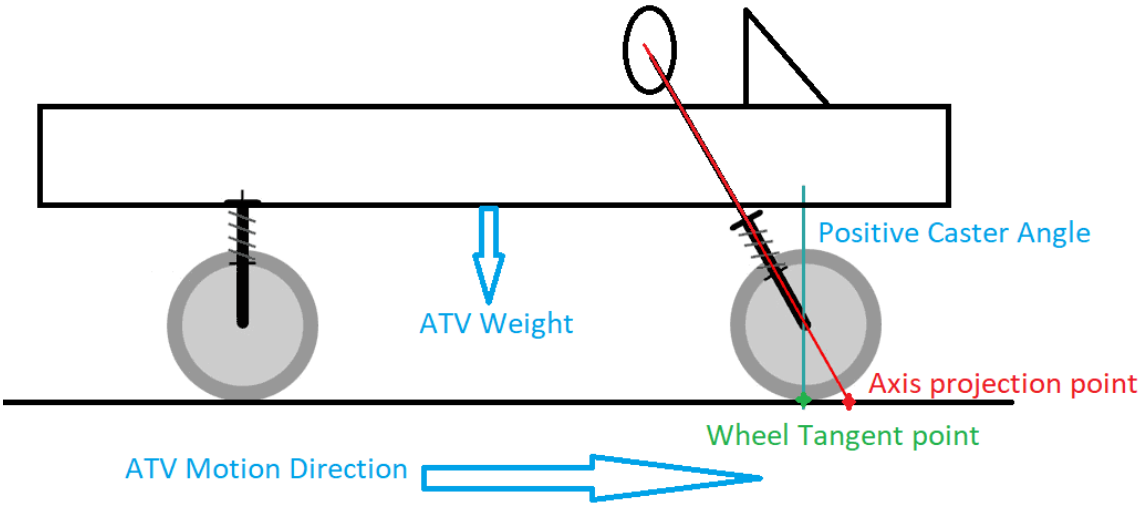

Figure 3.23: Positive Caster Angle for the vehicle direction.

is to make it easier for the ATV to maintain a straight line position by making the stable mechanical position for the front wheels in the center position while going forward. Figure 3.23 demonstrates that the projection of the axis is in front of the tangent point of the wheels on the ground. This position makes the wheels follow the axis projection. Whenever the steering wheel moves to the left or the right, the ATV weight will act as a force that pushes the steering wheel back to the center position.

This phenomenon is not the case while going backward. When going backward, the projection of the axis on the ground is behind the wheels' tangent point to the ground. This position is not a stable mechanical position for the wheels and leads the wheels to try to go behind the axis' projection. Consequently, when the steering wheel moves a little bit away from the center position, the ATV weight will help the steering wheel move more. Figure 3.24 demonstrates the Negative Caster Angle.

Another test was made to prove The Caster Angle reason theory. The test was to manually move the steering wheel slightly to the right when the ATV runs backward while disconnecting the steering module controller. As a result, the steering wheel

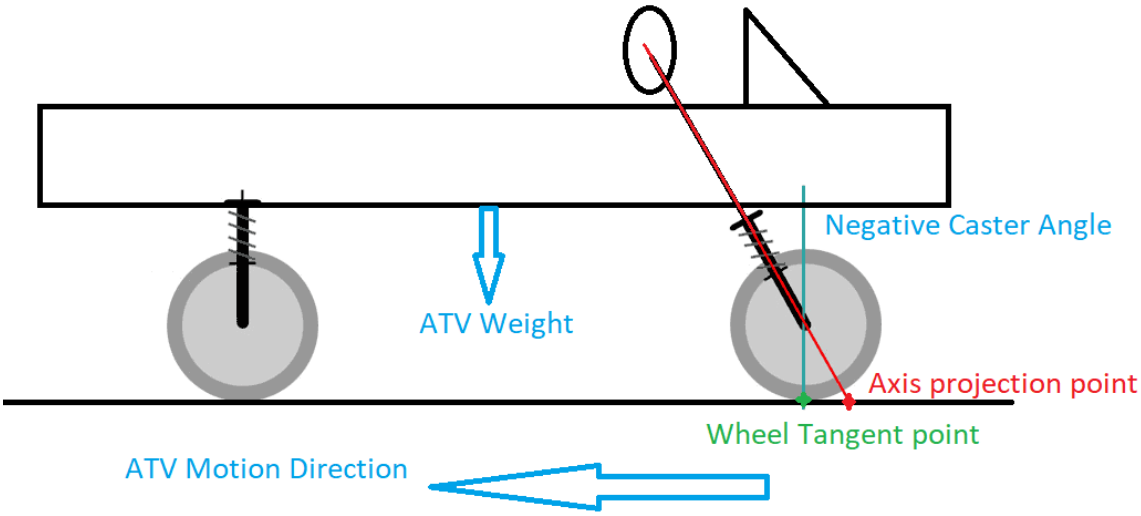

Figure 3.24: Negative Caster Angle for the vehicle direction.

went to the maximum right position. This result proved that the stable point for the wheels while going backward is behind the projection point of the axis on the ground.

As a conclusion to this phase, for the sake of the steering angle, the ATV should be used backward, or the Caster Angle should be adjusted to be neutral Caster Angle, or the EPS motor should be replaced with a more powerful motor, or apply external torque to the handlebar similar to the human driver.

Because the next step will be about an ML model where a human needs to drive the ATV to gather data using a camera mounted on the ATV, the backward solution is not practical. Also, because this research is about not to change anything in the ATV itself, the idea of replacing the EPS motor or adjusting the Caster Angle is not acceptable.

As a solution, an external motor is deployed with a pulley and an aluminum wire to mimic a human driver. Below are more details about the new design and implementation. The design has a mechanical part, electrical and hardware part, and software.

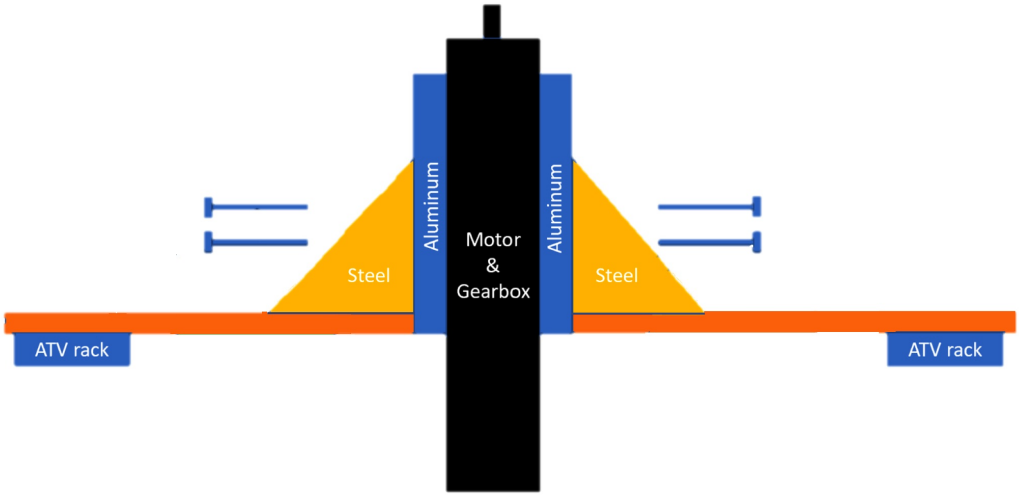

Figure 3.25: Motor mounting design

3.2.4.1 Mechanical Design for the New Steering Module

The primary approach of the design is to have a robot similar to human arms to move the handlebar. A motor Andymark am-0255 with a stall torque of 343.4 in-oz is mounted at the rear of the ATV. The motor is connected to a gearbox am-4008_020 with a gear ratio of 20:1. Researchers machined a pulley and mounted it on the top of the gearbox. A metal cable connects the two terminals of the handlebar, with the two sides crossing each other and turning around the pulley. In this way, the new motor controls the handlebar with the help of the EPS motor. The handlebar needs 50 lb-force to move it when the ATV is stationary, making this motor a matching choice.

The module is mounted to the ATV using two aluminum pieces surrounding the motor. An L-beam steal fixes the two aluminum pieces on an aluminum slab. Four stainless steel bolts, washers, and knobs fix this structure on the ATV chassis as illustrated in Figure 3.25

Figures 3.26 and 3.27 show the motor with the gear box and pulley mounted on the ATV.

The motor moves the handlebar using a metal cable. The cable crosses itself to

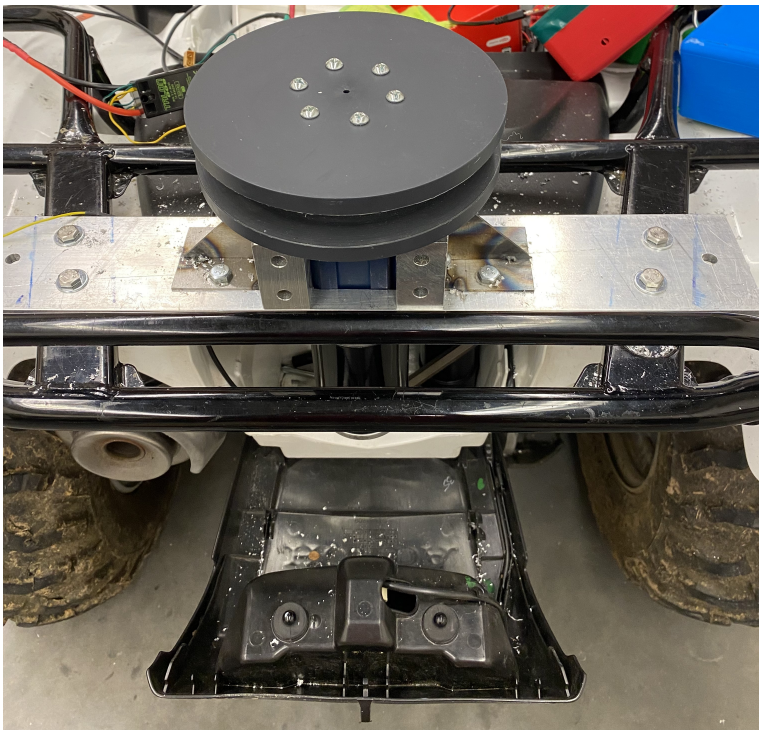

Figure 3.26: Motor mounted on the ATV

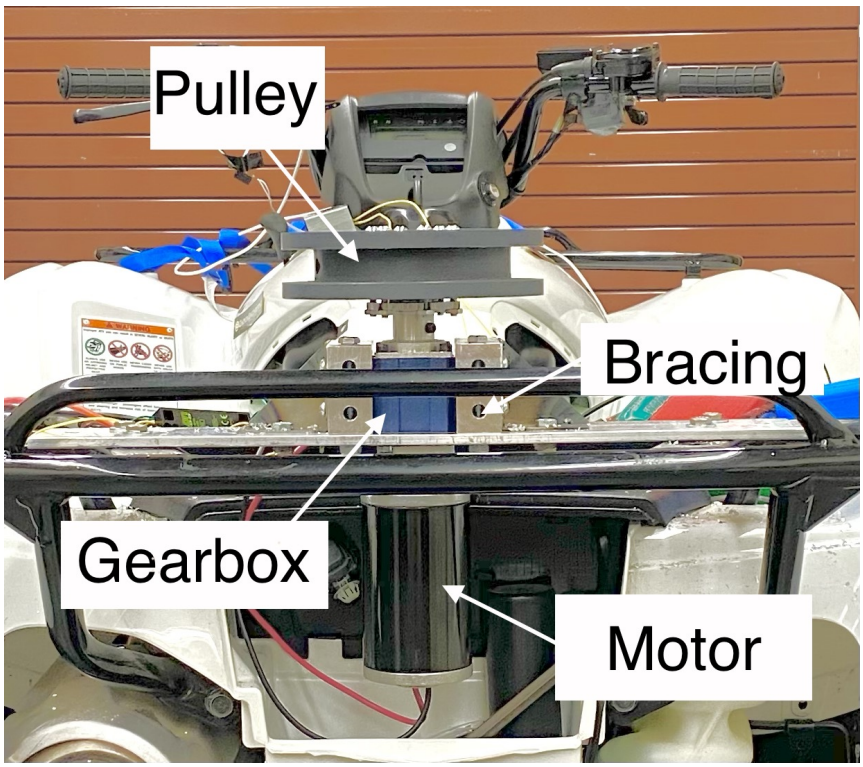

Figure 3.27: Motor mounted on the ATV - rear view

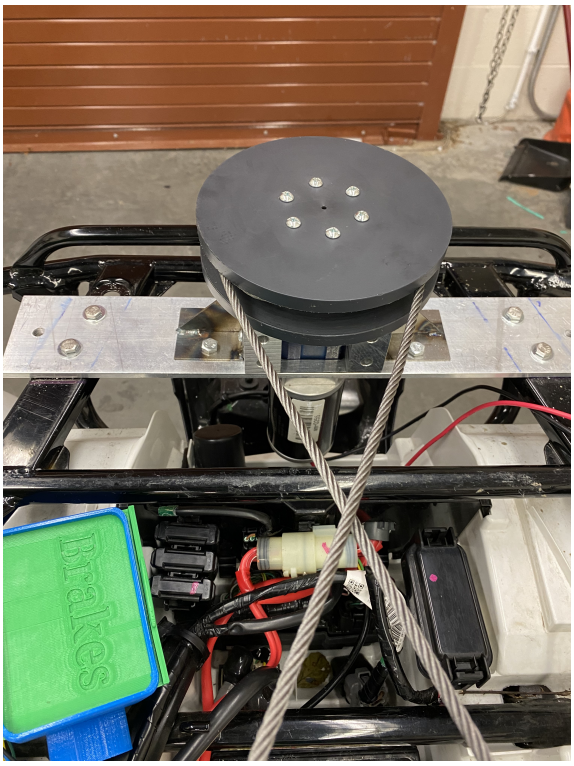

Figure 3.28: Cable driver of the handlebar attached to the pulley in a cross position to increase the range of motion

have a better angle with the handlebar to reach the maximum range of angles the handlebar can turn. Figure 3.28 demonstrates the cable on the pulley. Figure 3.29 show how the cable connects to the two terminals of the handlebar.

3.2.4.2 Hardware Design and Implementation of the New Steering Module

The motor requires a current between 2.7A to 28A to run. This current needs a motor driver to supply. The motor driver used is am-2854, as shown in Figure 3.30. The motor driver receives a PWM signal from the MSP430. The motor power equation is stated below. The electrical power needed should be equivalent to the mechanical output power.

Figure 3.29: Cable driver attached to the handlebar

$$P = \frac{RPM \times T}{5252} = VI, \text{assuming no losses, where T is the torque}$$

On the other side, a potentiometer is connected to the steering shaft to be able to measure the current steering angle using an ADC.

3.2.4.3 Software and Control Model for the New Steering Control System

The motor driver has a built-in PID controller. The driver is configurable to receive commands via CAN bus or PWM signals. In this research, a PWM signal with a duty cycle between 7% and 9% controls the driver, where 8% will make the motor neutral, and increasing or decreasing the duty cycle from 8% will make the motor rotate faster in the two directions. The steering angle controller needs feedback from the steering

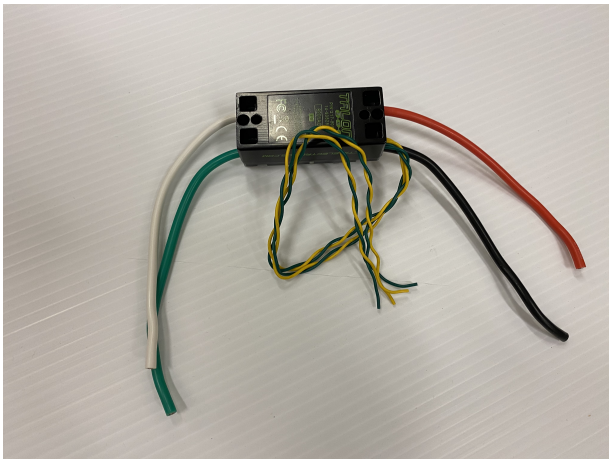

Figure 3.30: Motor driver used to control the new steering motor

shaft with the current steering angle. A 12-bit ADC reads the steering angle's value as the feedback. The microcontroller receives a CAN message with the required angle from the GPU. The software compares the required angle value with the current value from the ADC and adjusts the PWM duty cycle value to move the motor toward the correct angle.

When the engine is running, the motor and the EPS system can follow the command angle with high precision of 99.3%, which was a designed acceptable accuracy to maintain the steering wheel stable and resist the electric noise read by the ADC.

CHAPTER 4: INTEGRATION AND TESTING OF THE ATV BASE ARCHITECTURE

Chapter 3 discussed the implementation and design of the base architecture of the ATV. This chapter describes the integration and testing of the modules. Most test cases were executed twice, with the ATV suspended on four automotive stands to check the functionalities of the actuators and on the ground to test the environmental effects on the controllers. Figure 4.1 shows the ATV suspended on stands.

The CAN bus power wires are connected to the ATV battery. Each module has its MSP430 connected to the CAN Shield using a PCB. The CAN Bus power supplies this PCB, which converts the 12V to 5V. The PCB has a place to mount the MSP430 and the CAN Shield. Figure 4.2 shows the PCB design Schematic. Figure 4.3 shows the PCB layout. Figure 4.4 shows the manufactured board, while Figure 4.5 demonstrates the MSP430 connected to the CAN shield using the fabricated board. It passed all testings.

3D-printed boxes similar to Fig 4.6 contain the nodes to protect against the forest environment.

The GPU is not included in this phase. Another MSP430 acts as the central processing unit that provides commands to the actuators. The following chapters will discuss GPU integration.

4.1 Safety

For safety, the researchers designed and equipped the ATV with two kill switches, a wireless kill switch with a range exceeding 100 ft, and a hard-wired switch. These switches are for emergency use to stop the ATV.

Figure 4.1: The ATV suspended on stands

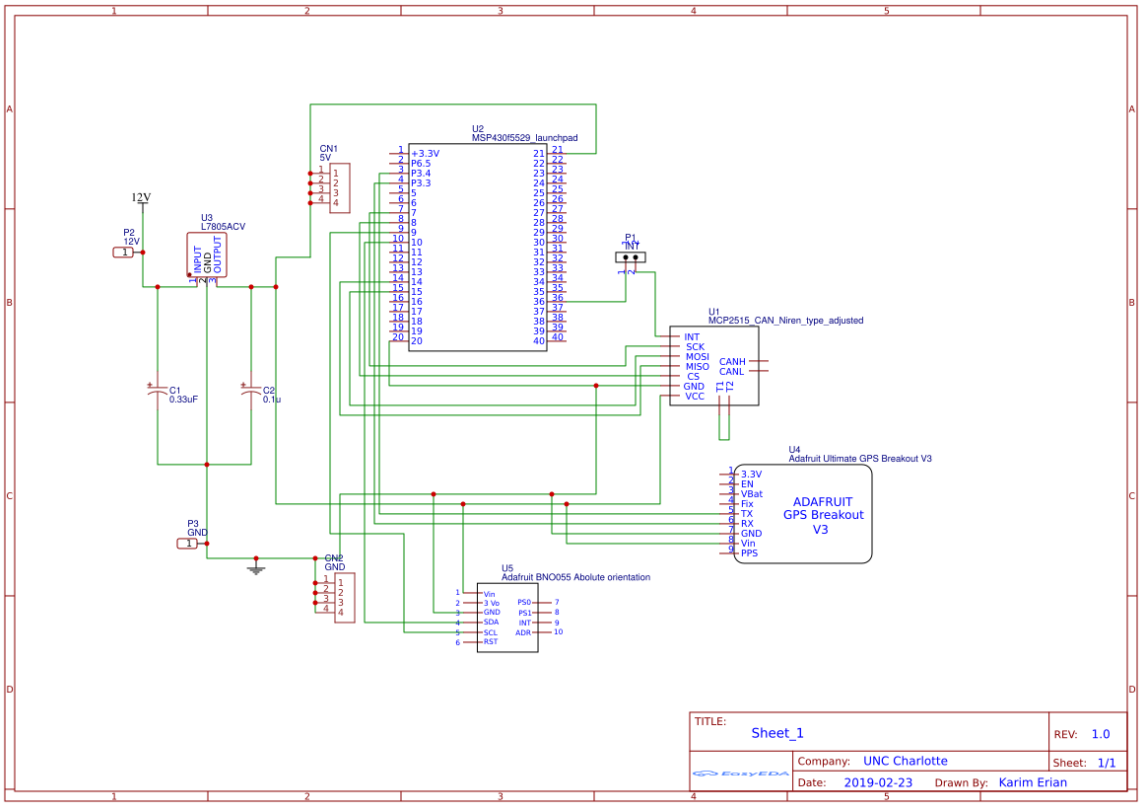

Figure 4.2: PCB design schematic

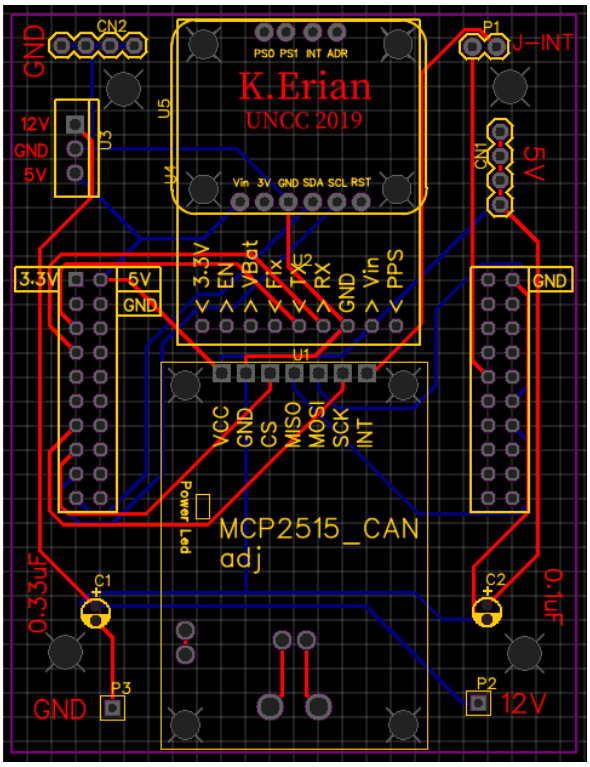

Figure 4.3: PCB layout

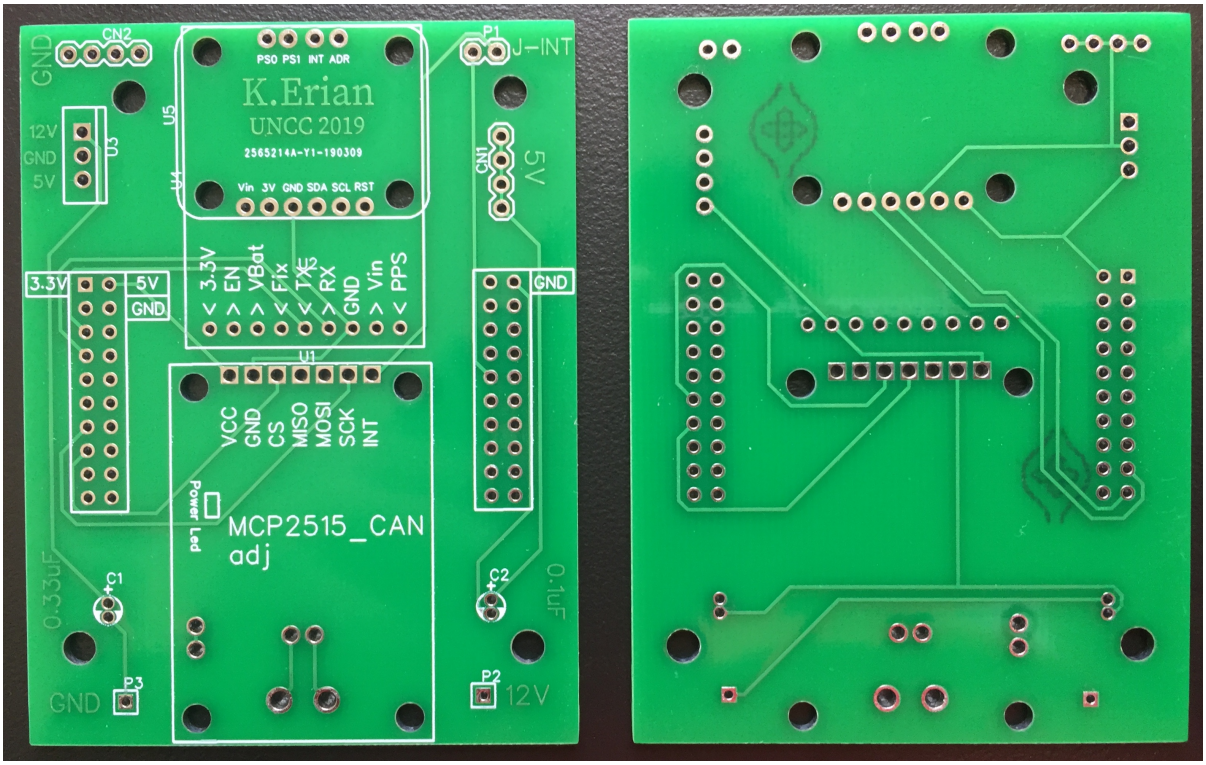

Figure 4.4: Fabricated PCB

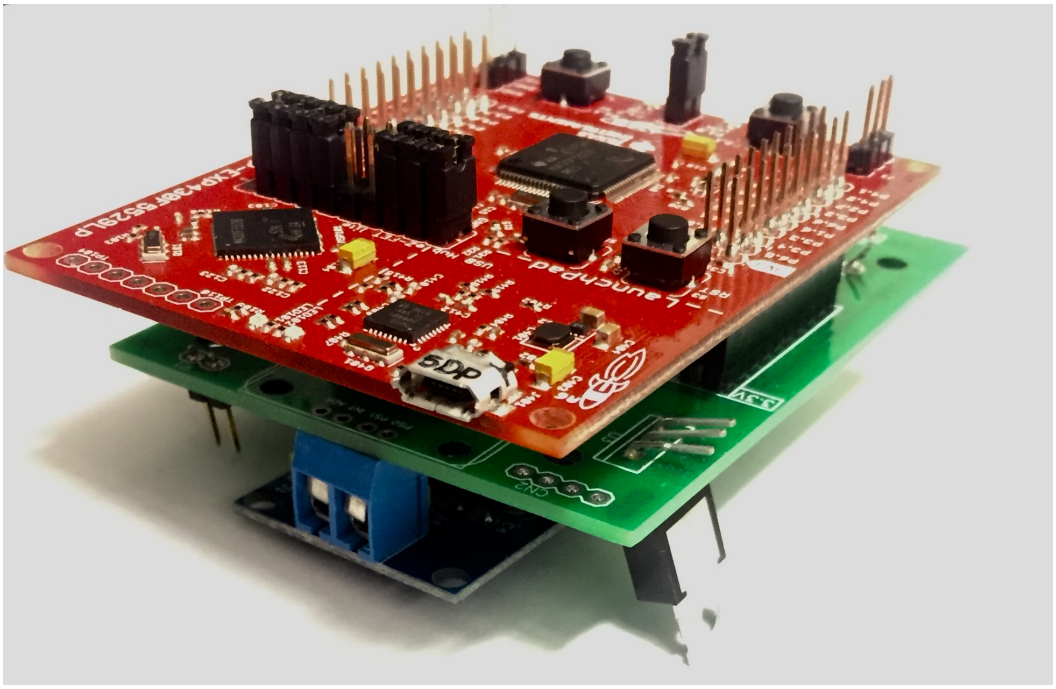

Figure 4.5: MSP430 connected to CAN Shield through the fabricated PCB

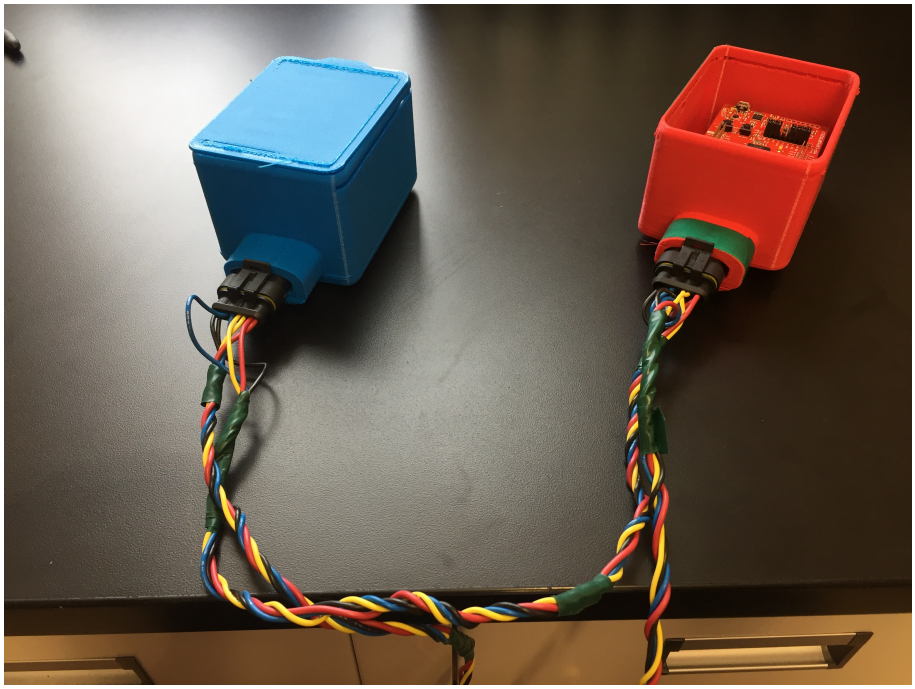

Figure 4.6: The 3D printed boxes connected using CAN bus

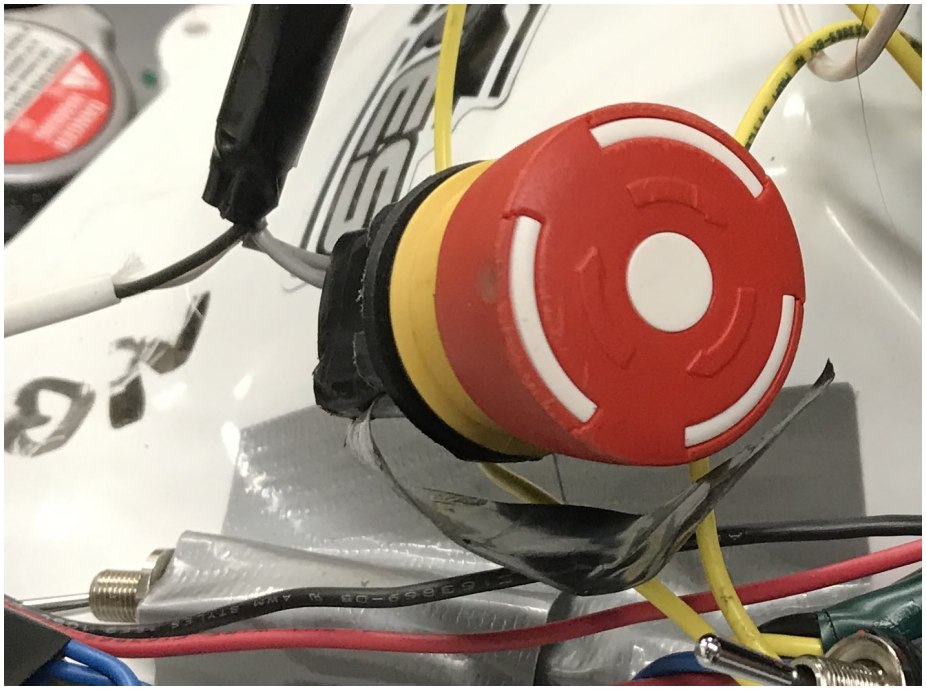

Figure 4.7: The ATV hard-wired safety kill switch.

Figure 4.7 shows the hard-wired switch. Wires connected the switch to the engine ON/OFF switch. The ATV switch must be in the OFF position to use any two switches.

The wireless switch is the same wire connected to a relay attached to a wireless receiver. Figure 4.8 shows the wireless receiver attached to the ATV. A hand-held remote controls the wireless receiver, which controls a relay attached to the wired kill switch.

4.2 Speed Control Module Design, Integration and Testing

The first design for the speed module was open-loop throttle control. Testing started by sending different duty cycle values from the central processing unit to the throttle controller. For the next test, the central unit sent the values to the throttle controller with the controller attached to the ATV's pulley controlling the air valve. The pulley rotated with the correct angles.

The researchers applied the test on the ATV while suspended on stands, on flat

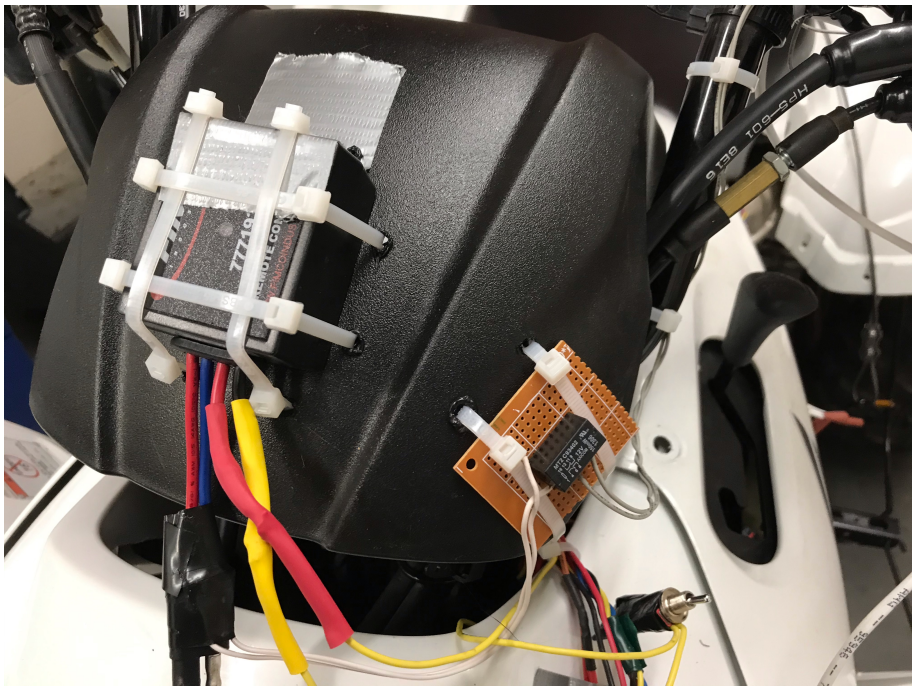

Figure 4.8: The ATV wireless safety kill switch.

ground, and in a forest environment. The ATV tires rotated at a speed that reflected the duty cycle for the first two cases. While testing in the forest, the ATV speed varied, reflecting the land topography and slopes when the duty cycle remained the same. Therefore, the researcher concluded that the throttle control is not robust enough, and the ATV needs a speed control with speed feedback. The research started with implementing a GPS model. The GPS provides the current speed as one of the GPS readings. The first test for the GPS speed reading was in a moving car. The GPS read the correct speed value. The researcher designed a code that compares the current speed from the GPS with the required speed and changes the duty cycle value accordingly. Testing this implementation concluded that the GPS data update rate is slower than needed leading to speed overshooting. Also, the GPS could not acquire satellites signals in the forest environment. Another method to measure the speed is the ATV velocity sensor. The velocity sensor allowed the ATV to maintain the same speed in a forest environment with different topography. There was no overshooting for a speed of 6 mph or higher.

4.3 Braking System Integration and Testing

The researcher integrated the braking system by mounting the linear actuator on the ATV and attaching it to the rear brakes pedal using the pins mentioned in chapter 3. The CAN bus supplies the braking system with power and connects it to the central processing unit. The following test was run twice, once with the ATV stationary, then with the speed control system moving the ATV. The braking system response time from sending the command till the brakes are fully applied is 1.1 seconds. The braking system starts to slow down the ATV within 250 ms. If the ATV travels at 6 mph (2.68 m/s), the ATV will move 67 cm before it starts to slow down and reach a complete stop within 1.2 meters on average. This value may vary with the path inclinations from 1.03 m to 1.58 meters. However, as mentioned in the speed control section, the central unit sends the stop command as a zero speed value, stopping the throttle servo motor and applying the brakes.

If adding the processing time of the GPU to make a stop decision, running SegNet at 24 fps with a reaction time of 50 ms, the ATV will reach a complete stop in 0.45 seconds. The average human reaction time is 1.19 seconds in a simple situation and 2.43 seconds in a complex situation [58], leading to a stopping distance of 3.7 meters to 7 meters for cars on standard road [59].

The central unit sent a command to release the brakes over the CAN bus, and the braking system worked as expected. The braking system response time from sending the command till the brakes are fully released is 1.7 seconds.

4.4 Steering Module Integration

In Chapter 3, the first integrated steering module was described. When testing this module on the ground with the speed module and the braking system, the ATV turned with a radius of 15 meters while going forward. The ATV was able to turn in a circle with a radius of 3 meters going backward. The ATV successfully did a

Figure-8 with a 3 meters radius for each circle and stopped at the same starting place while going backward. The gathering data for the ML model needed the ATV to move forward for a human driver to drive the ATV. Another reason is that the ATV should learn from the experienced driver driving forward and practice what the ATV learned. Therefore, the researcher designed and implemented the new steering module mentioned in Chapter 3. The new module could turn the handlebar in the full range while the ATV is standing still with the engine running. The EPS system helps the new motor to achieve full-range turning. The ATV made a circle forward with 3 meters radius using the speed control system and the new steering model.

CHAPTER 5: MACHINE LEARNING SYSTEM DESIGN AND PERFORMANCE GOAL

This chapter is about the AI models used to train the ATV to follow a path in the forest. This chapter also discusses a proposed method to measure the system performance using AI methods.

5.1 Controlling the ATV using AI Models

Executing algorithms for autonomous vehicle operation has shown success in typical situations. However, while implementing algorithms, it appears that 35,000 use cases are still not representing a sufficient number of situations that can happen in real life [3]. Those situations vary from a simple "moving straight forward in a lane then a light signal turns red" to a more complicated one like "a left turn into another street where a ball is rolling in the middle of the street and a child running after it". Self-driving cars algorithms were showing success for every added scenario. However, if the designer does not conceive the scenario, then it is not solved. Therefore, observing human driving behaviors to learn from them would be a better solution. Machine Learning, in this case, seems to be the correct answer.

Because this research is about following a path in the forest using a camera, the next step in this research needs the ATV to identify a paved path and identify its position on the path. The step after is to understand how an experienced driver, with highly desirable steering action, would steer the ATV to allow the system to learn from the human driver [11, 60]. Therefore, to identify the path and learn from an experienced driver, data gathering is mandatory.

The process to achieve the final goal is summarized in Fig. 5.1. As illustrated in the

Figure 5.1: Implementation Process Summary

figure, data gathering is the first step in order to gather data to understand the ATV location with respect to the paved path and data to understand the normal human reaction in each location. Then the data should be prepared and preprocessed to the proper format for each model used. Then path identification eventually is made using semantic segmentation methods, and the last part of the process is to use a machine learning regression model in order to identify the proper action to be made by the ATV, in this case, what steering angle should be applied to the handlebar to stay on the path.

5.1.1 Data Gathering

There are two different kinds of data needed: the first is the visual data from the camera, and the other is the angular values from the steering wheel. The angular data gathering used an encoder attached to the steering axis. This encoder is providing analog values between 0 and 3.3V. An MSP430 microcontroller with ADC reads these values and sends them to a laptop that saves them in a text file.

The visual data gathering uses an Intel Deep Sense i435 camera mounted on the ATV. This camera has a wide-angle RGB 16MP camera, 2 IR sensors, LASER Depth Sensor (LIDAR), and an IMU with gyro and accelerometer. Fig. 5.2 shows the camera mounted on the ATV. An experienced human driver operated the ATV on the paved path in the woods behind the UNC Charlotte EPIC building, as shown in Fig. 5.3. This experienced driver's steering actions are highly desirable steering actions. Fig. 5.4 demonstrated the driver operating the ATV in the center of the path for path

Figure 5.2: The cameras mounted on the ATV

detection and as an example of good driving. The visual data gathered consisted of six trials; all videos included the steering angle data recorded concurrently.

5.1.2 Data Pre-processing

The first step was to align the ADC samples with the camera frames. To do that, a decision to visualize the values of the angles with the captured video seemed to be the best idea to align them together. The visualization process uses iMovie software from the Mac OSX to align all videos with the steering angle. The process starts with MATLAB making a video of a rotating arrow with the angle value from the data collected. Then iMovie overlaps this video with the video of the paved path from the camera. Mainly iMovie was used to apply changes in the rotating arrow angles video to the video speed, start, and end. Those changes were applied until the

Figure 5.3: The route in the woods from google maps. Different routes were from B to A, B to C and opposite direction.

Figure 5.4: The ATV driven in the center of the path

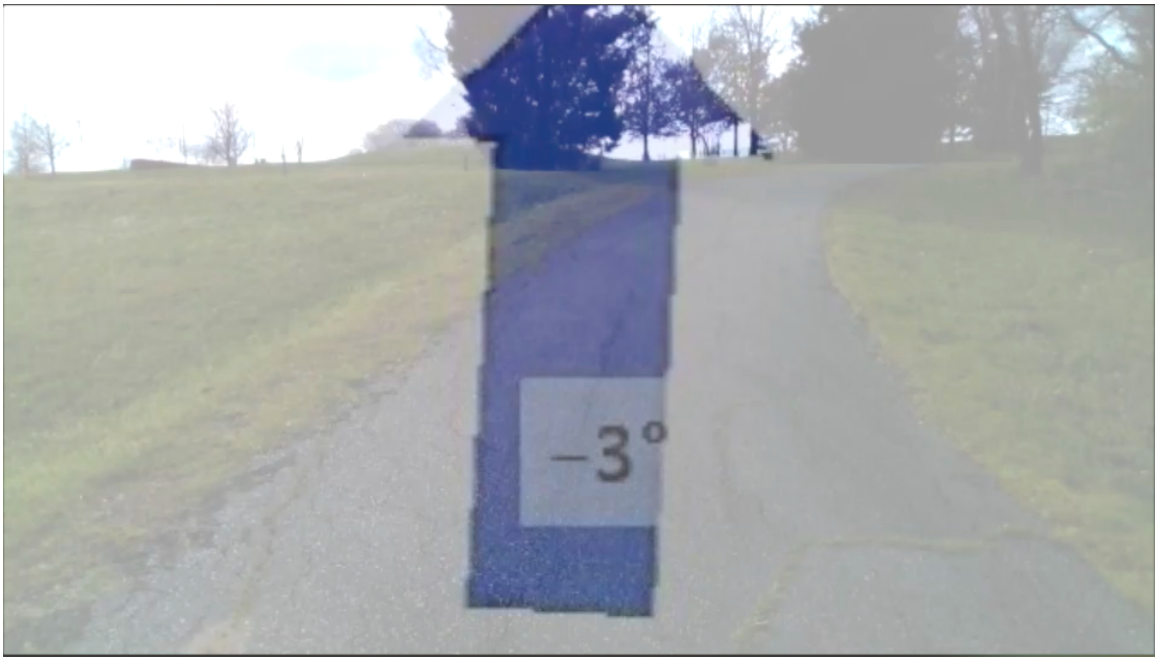

Figure 5.5: Visualization of angles with image

rotating arrows matched all path turns in the camera videos. Fig. 5.5 illustrates an overlapped frame.

Furthermore, after synchronizing the videos, the start and end were trimmed to remove the parts of videos that do not match.

Finally, the last step of data preprocessing was to apply a down-sampling technique to have the number of angle samples match the number of video frames per second. The timing was taken into consideration while down-sampling. The problem was that the angle data were recorded at 90, 97, and 112 samples per second, whereas the camera videos were 30 fps. While 112 and 97 do not have a common factor of 30, the technique applied a particular pattern of sample dropping. This downsampling technique allowed the number of angle samples to match the camera video frame rates of 30 samples per second.

5.1.3 Path Detection

This section illustrates different methods used to detect the path and their results. The difference between a path in the forest and the city roads is that the forest path

Figure 5.6: Sample of the path to follow in the forest

is not a standard road with a fixed width, and it does not have any painted lines on it. Also, this path has some challenges like shadows, different light intensities, different light distributions due to interleaving trees, cracks, and light reflection on the path. Fig. 5.6 illustrates the paved path used for data collection with various lighting and shadows.

5.1.3.1 Canny Edge Detection

The first trial used simple edge detection. The edge detection mechanism is similar to a high-frequency filter that detects the sudden change in the image colors' value. The implementation used a code from the Mathworks repository and applied it to an input RGB image. It follows the steps [61]:

1. Convolution with Gaussian Filter Coefficient
2. Convolution with Canny Filter for Horizontal and Vertical orientation
3. Calculating directions using atan2
4. Adjusting to nearest 0, 45, 90, and 135 degree
5. Non-Maximum Suppression

Figure 5.7: The path using color masks

6. Hysteresis Thresholding

When applying this technique to the image in Fig. 5.6, the output image had all the edges except the path's edges. Therefore, this approach was deemed unsuccessful.

5.1.3.2 Color Filters

After observing the Canny edge detection results, the next approach was filtering the image by colors. The Matlab Color Threshold app auto-generated code by implemented different masks. The best results were when the filters from RGB representation with HSV and LAB were combined. The results were more promising than the output of the Canny Edge. Fig. 5.7 illustrates the results. This solution showed acceptable results compared to the Canny Edge detection.

5.1.3.3 Combining Color Filter with Canny Edge Detection

Now having the path more evident in the image, the following approach applied the Canny Edge detection the colored masked and combined with the original image to find the path separated, as shown in Fig. 5.8

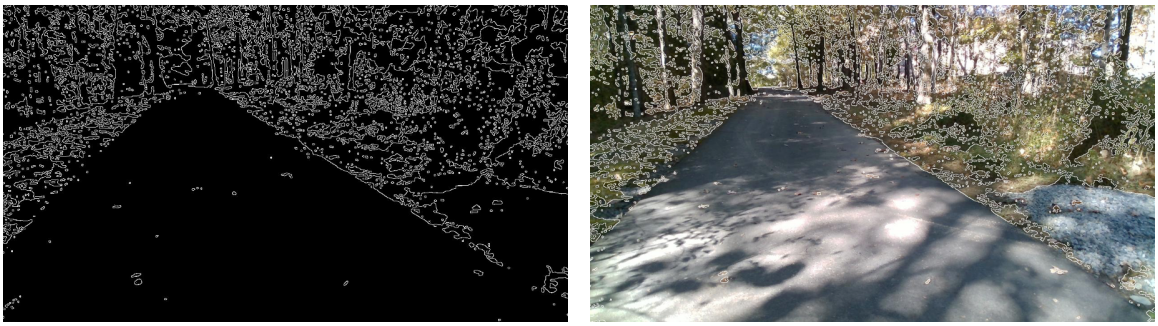

Figure 5.8: Applying edge detection on the color masked frames

5.1.3.4 OpenCV Semantic Segmentation using CityScape

Cityscape [62] is a dataset with data from many different cities around the world. The dataset consists of around 3700 labeled images. It contains 20 classes. Fig. 5.9 shows the network components.

In the next trial, Efficient Neural Network (ENet) light network design was used [63]. ENet can process 0.25 frames per second to detect 20 classes. The results were in the 90s percentile when only detecting four classes (roads, sky, trees, and vegetation). The other classes are not necessary for this context as they would not appear in the woods. Fig. 5.10 illustrates the output.

ENet is using Bottleneck layers instead of Convolutional Layers, according to [63], it is 18x faster than an average CNN. The pretrained network is a light model of 3.4MB compared to Nvidia semantic segmentation for road detection. The next one tried was 1.1 GB.

The processing time was decreased to 0.94 seconds per frame on average (approximately 1.1 fps) using a standard MacBook Air with only four classes. The ENet programming is not CUDA-enabled, which means using a GPU will not make it faster.

5.1.3.5 SegNet ResNet18 DeepScene Semantic Segmentation

The research considered another model to optimize the SS computation time: SegNet, also called the FCN-ResNet18-DeepScene S.S. model [64], which is trained on

Table 1: ENet architecture. Output sizes are given for an example input of 512×512.

Name	Type	Output size
initial		$16 \times 256 \times 256$
bottleneck1.0	downsampling	$64 \times 128 \times 128$
$4\times$ bottleneck1.x		$64 \times 128 \times 128$
bottleneck2.0	downsampling	$128 \times 64 \times 64$
bottleneck2.1		$128 \times 64 \times 64$
bottleneck2.2	dilated 2	$128 \times 64 \times 64$
bottleneck2.3	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.4	dilated 4	$128 \times 64 \times 64$
bottleneck2.5		$128 \times 64 \times 64$
bottleneck2.6	dilated 8	$128 \times 64 \times 64$
bottleneck2.7	asymmetric 5	$128 \times 64 \times 64$
bottleneck2.8	dilated 16	$128 \times 64 \times 64$
Repeat section 2, without bottleneck2.0		
bottleneck4.0	upsampling	$64 \times 128 \times 128$
bottleneck4.1		$64 \times 128 \times 128$
bottleneck4.2		$64 \times 128 \times 128$
bottleneck5.0	upsampling	$16 \times 256 \times 256$
bottleneck5.1		$16 \times 256 \times 256$
fullconv		$C \times 512 \times 512$

Figure 5.9: ENet Architecture [63]

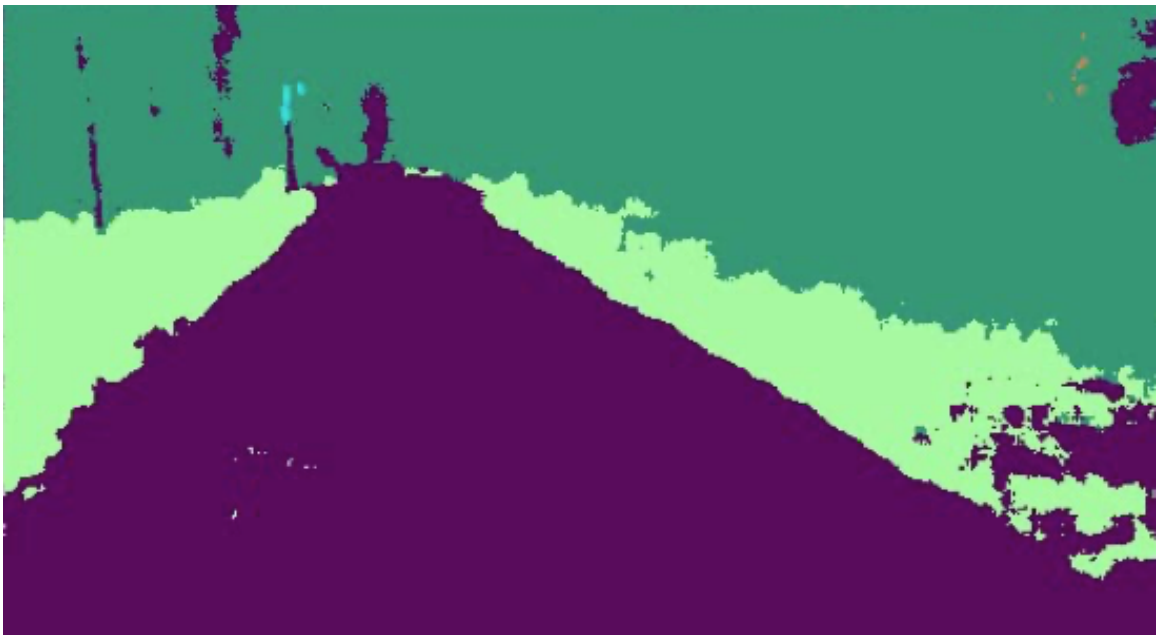

Figure 5.10: ENet output

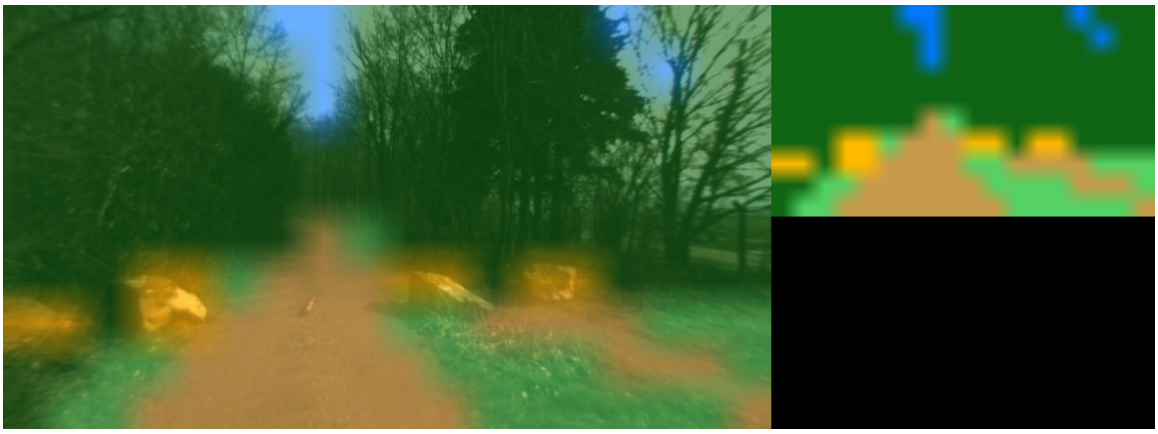

Figure 5.11: Example for the SegNet S.S. [64], on the left it is the overlay of the mask on the original image, while on the right, there is the mask used

ResNet18. An Nvidia team created this model for the Jetson hardware family. The SegNet DeepScene model used the Freiburg Forest data set [5]. The data set was collected in 2016 by a robot with cameras wandering around the Freiburg Forest. This model has two options: a higher resolution and a lower resolution option. The higher resolution is 864x480 and the lower resolution is 576x320. The advantage of this model is that the Nvidia team designed it for the Jetson family GPUs. The higher-resolution model realized a processing speed of 14 fps with GPU time per frame varying between 84 and 90 ms. The lower resolution model was even faster and had a rate of 24 fps, with a GPU time-varying around 45 to 50 ms per frame on the Jetson Nano. The model is set only to classify paths, vegetation, and trees. This setting did not have any adverse effect on results.

Fig. 5.11 illustrates the overlayed versus the masked output of the SegNet. Fig. 5.12 (LEFT) demonstrates the output mask for the low-resolution model, and Fig. 5.12 (RIGHT) illustrates the output mask for the high-resolution model for the same path. The lower resolution seems to be smoother than the higher resolution in this case.

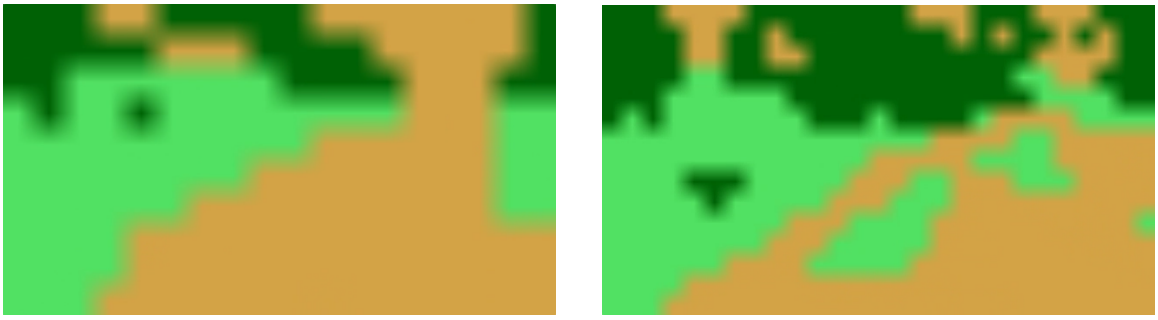

Figure 5.12: left: SegNet Low resolution output, right: SegNet High resolution output

5.1.3.6 Method Used to Detect the Road and the Decision Taken

The use of SS methods is more effective and reliable than the classical image processing methods mentioned above, especially in contour detection.

The choice of an SS method considered two significant features: the precision of finding the path and the real-time consideration. The model's precision calculation used 21 samples as the ground truth. Fig. 5.13 illustrates one of the 21 samples used as ground truth for the semantic segmentation. The precision calculation of finding the path compared the path area of the masks of ENet with the ground truth. Then it compared the path area of the masks of ENet with the ground truth. It also adjusted the ground truth resolution to the mask's resolution to find the matching percentage. It is critical to predict a true path. This information is essential because if the model has a false positive, like detecting a path while it is a tree, then the ATV will crash. However, if the model did not detect all the paths in the image, but the path detected is true, this makes no safety issue even if it makes it harder for the ATV to follow. Considering this criterion, the ENet has more precision, but the precision of the SegNet is still close to the ENet.

For a real-time consideration, the SegNet is much faster as the SegNet is CUDA-enabled and can use the GPU to have a parallel computation. The SegNet can do 24 fps while the ENet is 1.1 fps. The SegNet Low-Resolution model can process a frame in less than 50 ms (41 ms and assume 9 ms for the software's other calculations

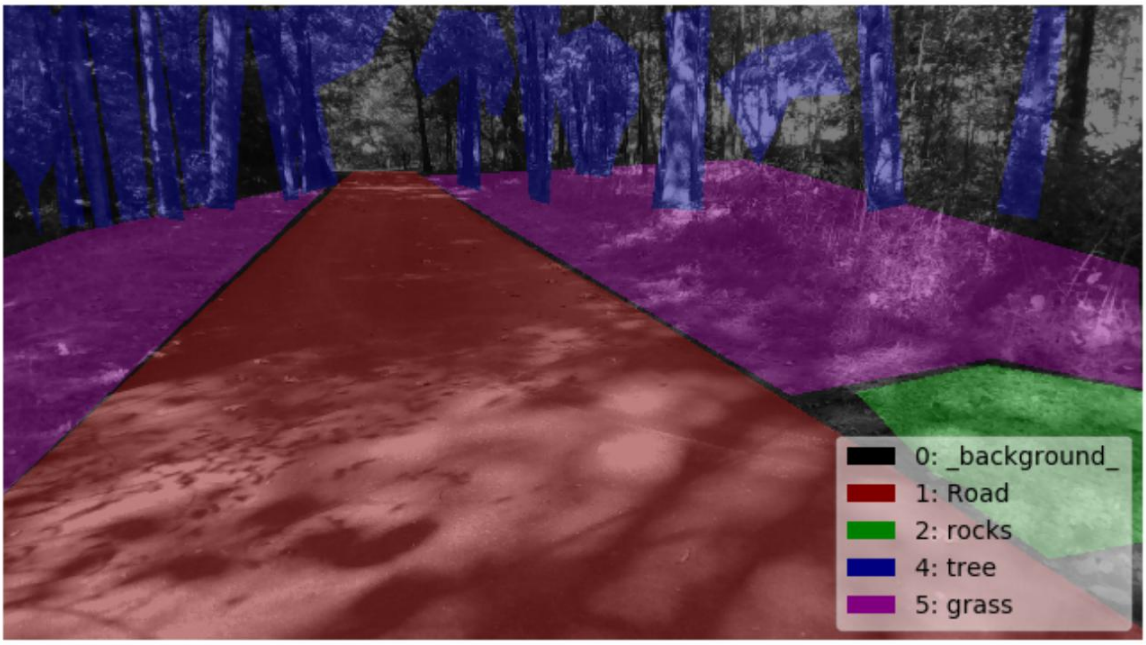

Figure 5.13: A sample of the ground truth for the S.S.

than SS), while the ENet can do a frame in 0.9s in addition to 9 ms for the other calculations making it 0.909 seconds. If the ATV moves at 6 mph (2.68 m/s), the ATV can see something new after traveling for 13.4 cm using the SegNet or 2.41 meters using the ENet.

With the ENet having higher precision but slower response time, the decision was made to use the SegNet to detect the path.

5.1.4 Preparation for the Machine Learning Models

The output of the SS is six videos. The training process used five videos, and the testing process used one, with a total of 33,316 frames for training and 7,743 frames for testing. Each SS frame has a corresponding steering wheel angle. The machine learning model needs the current ATV position on the path and the ATV trajectory if the ATV continues straight which are calculated from the SS frames. The ATV position is referred to as ATV current position or State 1, and the ATV predicted trajectory if the ATV continues straight forward is referred to as ATV next position or State 2 as illustrated in Fig. 5.14.

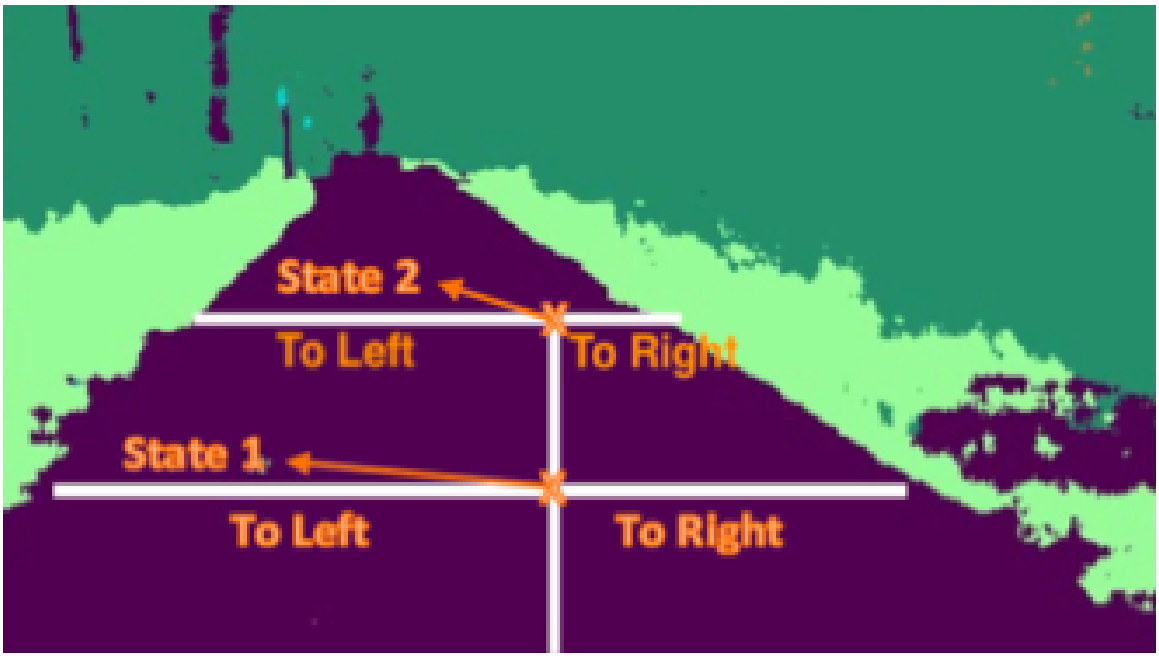

Figure 5.14: ATV current state (State 1) verses ATV trajectory projection state (State 2)

Seven sections divided the path to define State 1 and State 2, [-3,-2,-1,0,1,2,3] where "-3" is outside the path from the left side, and "3" is outside the path from the right side. 0 means the center of the path, the negative values are on the path to the left, and the other positive values are on the path to the right. Fig. 5.15 represents the different positions in details.

Each frame provided the two positions, State 1 and State 2, the software combined them with the corresponding angle value for the steering axis orientation. The data created two tables, one for training with a dimension of 33,316x3, and one for testing with 7,743X3, where two are the input data, and one is the ground truth to check the results.

5.1.5 Machine Learning Regression Models

A model can predict the angle value using the data table from the current position and ATV trajectory position. A regression model is appropriate for this prediction because the angle values are continuous. Since the data is simple, a linear model

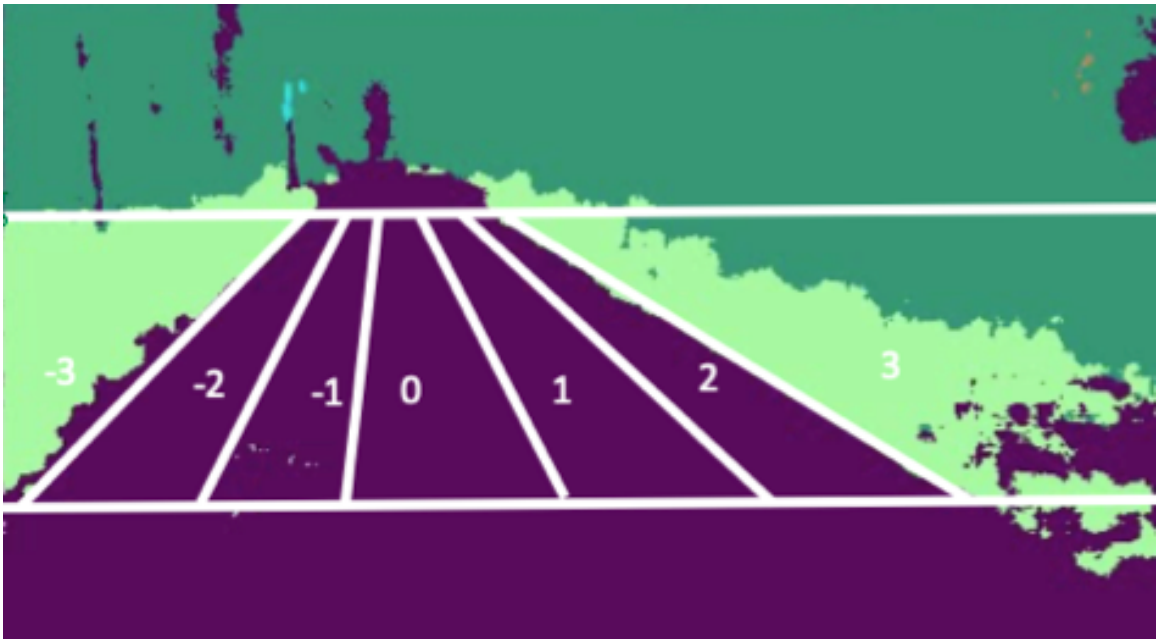

Figure 5.15: ATV position and state definition

should be sufficient.

The research considered two Linear Regression [65] models: the least-squares (LS) regression model and least-mean-squares (LMS). Let the training data (State 1 and State 2) be X, and the ground truth to be learned (the angle) is Y. We train the model to reach the equation $Y = WX$, where W is a weight and bias matrix. For the least-squares, we apply the equation: $W = (X^T \cdot X)^{-1} X^T \cdot Y$ to find the best suitable W that will create a linear equation between samples of X and predicted angle Y. For the Least Mean Square model, the equation is $W = W - \alpha \cdot Error \cdot X$ where the $Error$ is the difference between the predicted value WX and the ground truth Y. This needs multiple iterations until it converges. The main difference between the least-squares (LS) and the least-mean-squares (LMS) is calculating the error. The research implemented both methods, and the next chapter will discuss the results.

5.2 Measuring Driving Performance Using AI

This section discusses a proposed method to measure the performance of machine learning driving. The proposed solution starts with data gathering of an experienced

driver's behavior while driving at the center of a paved path in the woods, followed by preprocessing data to extract the video frames from the ROS file (the data gathered). An off-the-shelf light SS model finds the path and identifies the ATV states. Then, the Linear IRL model for small states space by Russel & Ng [14] defines the rewards per state from an experienced driver. The rewards help in deriving the rating function. The idea for the rating function is not to judge the future intentions by using a prediction model and compare the trajectory to the prediction. Instead, it gets the performance from the true path the ATV already passed through.

The ATV gathered seven different datasets for different driving behaviors following the same data-gathering technique. Matlab extracted the videos, and the research applied the SS model to get the data under test. A rating function evaluated the other drivers' performances.

This section will go through the different phases of implementation. Fig. 5.16 illustrates the proposed solution architecture. First, the ATV gathered the data. Then, the Rosbag output video is input to the ENet SS model. An IRL model - explained below - used the output from the ENet SS network in order to extract the system rewards. Then seven different driving attitudes were gathered as follows: right inside (Ri), right partially outside (Ro), left inside (Li), left partially outside (Lo), zig-zag inside (Zi), zig-zag inside and outside (Zo), and Random. The ENet S.S. used the videos as the input. The rating function used the SS output with the outcome rewards from the IRL as an input to obtain the performance rating for each driving attitude.

5.2.1 Data Gathering

The same Intel Deep Sense i435 camera used before was mounted on the ATV, as shown in Fig. 5.2. A human operator drove the ATV in the paved path in the woods behind the EPIC building shown in Fig. 5.3 and gathered 120 GB of data. The human operator drove the ATV in the path's center to detect the path as an

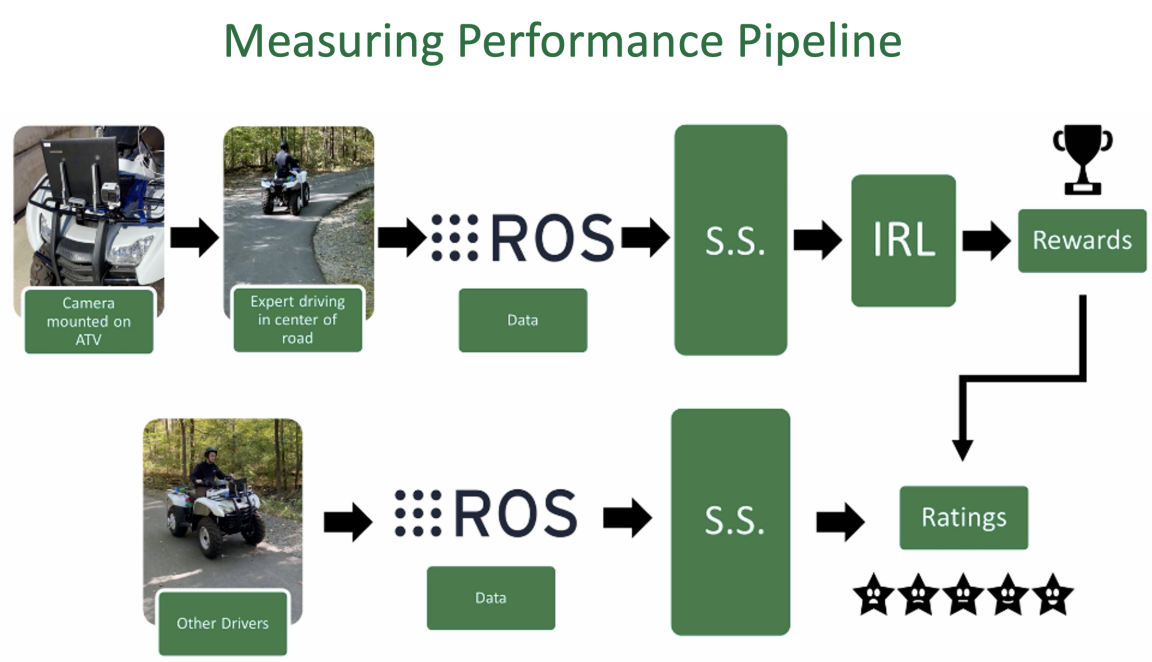

Figure 5.16: Rating driving method architecture

example of a good driving technique. Then they drove it at the right side of the path with all wheels inside (Ri), and with one wheel outside (Ro), same on the left side of the path (Li and Lo), then in a zig-zag shape inside the path (Zi) and another zig-zag while getting in and out of the path (Zo), and finally in a randomized way.

In this section (Measuring Driving Performance), the research is only concerned with the camera data as it cares about the results for the driving technique, not about the actions taken to drive.

5.2.2 Road Detection

In this section (Measuring Driving Performance), ENet discussed before is used. It has a better resolution, but it is slower than the SegNet. In the case of performance measurement, it is not a real-time requirement. Therefore, spending 0.9 seconds per frame is acceptable.

5.2.3 IRL Design and Implementation

The first action is to choose between Ziebart et al. [22] and Ng & Russell [14] and others mentioned in the background section above to implement the IRL model. The Ziebart et al. model is more advanced and complex than needed, as they are more focused after recovering the reward function about getting a similar optimum policy - maximum entropy and deep maximum entropy methods. Therefore, choosing to implement the IRL module per Ng & Russell paper [14]. Russell has two models for IRL depending on the state space size, and as there are only five states in this research, the small states space method for linear IRL was chosen, which is more accurate than the large-state space method. This model requires both the optimal policy and the transition probabilities for implementation.

5.2.3.1 The Mathematical Design and Implementation for IRL Model

Bellman equation is written in Matrix form as follows [14, 15]:

$$\boldsymbol{V}^{\pi^*} = \left(\boldsymbol{I} - \boldsymbol{\gamma}\boldsymbol{P}^{a^*}\right)^{-1}\boldsymbol{R}$$

In order to have an optimum policy:

$$\forall a \in \mathcal{A}\backslash a^*, \quad \boldsymbol{P}^{a^*}\boldsymbol{V}^{\pi^*} \geq \boldsymbol{P}^{a}\boldsymbol{V}^{\pi^*}$$

$$\forall a \in (\mathcal{A}\backslash a^*)\,. \quad \left(\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a}\right)\left(\boldsymbol{I} - \gamma\boldsymbol{P}^{a^*}\right)^{-1}\boldsymbol{R} \geq \boldsymbol{0}$$

Another constraint should be added to remove the trivial solution of $\boldsymbol{R} = 0$, which is to Maximize:

$$\sum_{s\in\mathcal{S}}\left(Q^{\pi^*}(s,a^*) - \max_{a\in(\mathcal{A}\backslash a^*)} Q^{\pi^*}(s,a)\right)$$

This means deviating from the optimum policy will reduce the value function and total rewards.

$$\text{maximize} \sum_{i=1}^{|\mathcal{S}|} \min_{a \in (\mathcal{A} \backslash a^*)} \left(\left(\boldsymbol{P}^{a^*} \right)_i - \left(\boldsymbol{P}^{a} \right)_i \right) \left(I - \gamma \boldsymbol{P}^{a^*} \right)^{-1} \cdot \boldsymbol{R} - \lambda \| \boldsymbol{R} \|_1$$

$$\begin{aligned} &\text{s.t. } \forall i \in 1, \ldots, |\mathcal{S}|. \forall a \in (\mathcal{A} \backslash a^*) \,. \\ &\text{-} \left(\left(\boldsymbol{P}^{a^*} \right)_i - \left(\boldsymbol{P}^{a} \right)_i \right) \left(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*} \right)^{-1} \cdot \boldsymbol{R} \leq 0 \\ &\text{and } \forall i \in 1, \ldots, |\mathcal{S}| \cdot |\boldsymbol{R}_i| \leq R_{\max} \end{aligned}$$

In matrix form: (M & u are dummy vectors for the sake of matrix dimensions)

$$\text{maximize} \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ -\lambda \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{M} \\ \boldsymbol{u} \end{bmatrix}$$

$$\begin{bmatrix} \left(\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a} \right) \left(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*} \right)^{-1} & -\boldsymbol{I} & \mathbf{0} \\ \left(\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a} \right) \left(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*} \right)^{-1} & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{I} & \mathbf{0} & -\boldsymbol{I} \\ \boldsymbol{I} & \mathbf{0} & -\boldsymbol{I} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{M} \\ \boldsymbol{u} \end{bmatrix} \leq 0$$

The python matrix solver finds R to recover the reward function/vector. The last matrix is modified to positive transitional probability to maximize the rewards for the most visited state by the agent.

The IRL model needs states, actions, and the transitional probability from one state to another by applying a specific action to run.

The states were given from the video and translated by the code. Five states were defined as shown in Fig. 5.17: Center, Right inside, Right outside, Left inside, and Left outside.

The code checks the color value for the middle pixel of the frame's x-axis and 1/3 from below the y-axis to determine the state. If it is the color defined as the path, then it is inside, therefor it checks the distance to the left and the right path

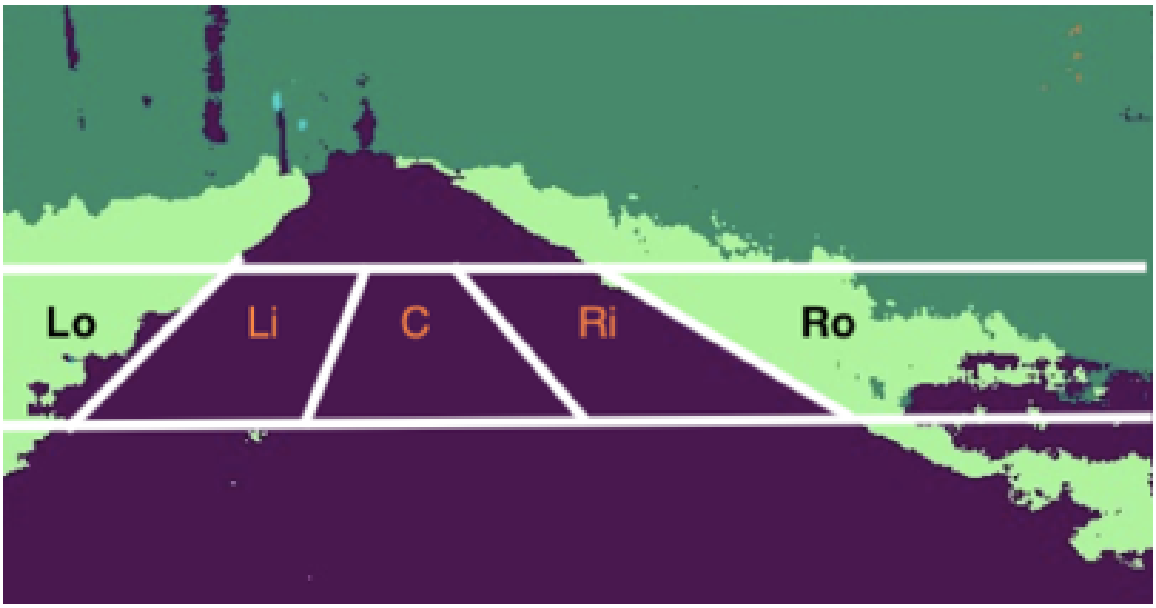

Figure 5.17: IRL states definition

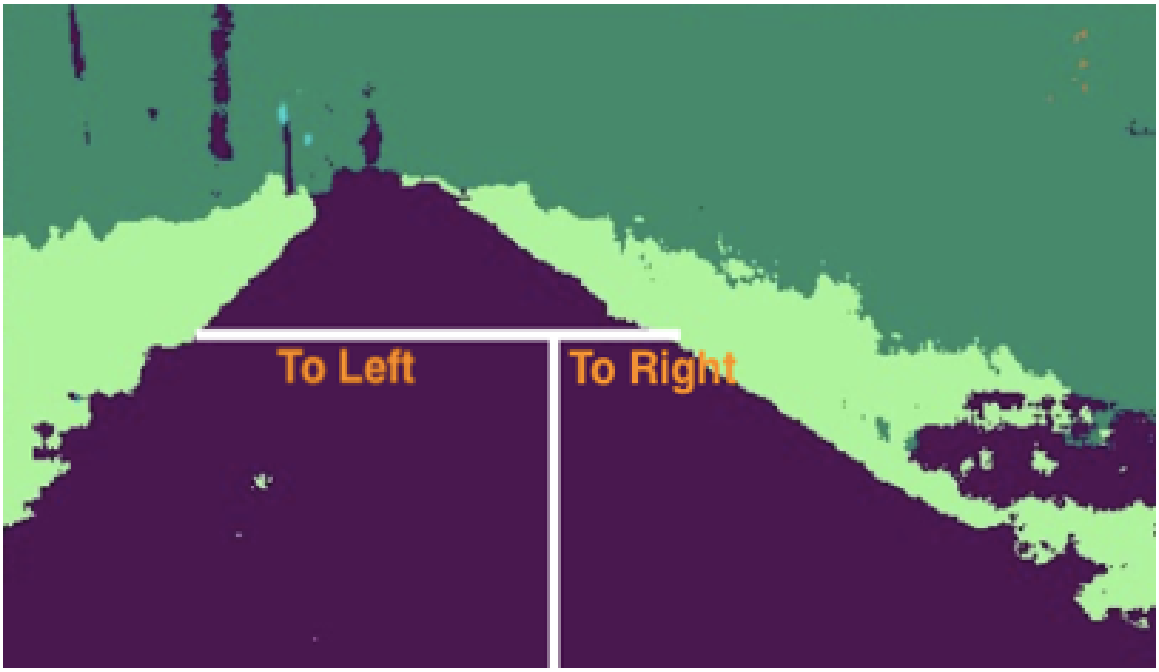

Figure 5.18: IRL states calculation

boundaries, and according to the number of pixels to the left and right, it gets the state. Otherwise, if the pixel's color was outside the path, the code checks for the previous state and determines the state. Fig. 5.18 illustrates the calculations of the states from the input frame.

The mathematical conditions for the three middle states are as follows: The following condition is for the center position (R: distance to the right edge of the path, W: is the path width in pixels):

$$R - \frac{2w}{3} < 0 \quad \& \quad R - \frac{w}{3} > 0$$

If the values do not fulfill the above condition, but they fulfill the condition below, that means it is the Left inside state:

$$R - \frac{2\omega}{3} > 0 \quad \& \quad R - \frac{\omega}{3} > 0$$

Otherwise, it is in the right inside state.

The actions were recorded in a text file and used as input to the model. The available actions are Left, Forward, and Right assuming that the ATV is always moving at the same speed and no need for brakes. The software calculates the transitional probability from the experienced driver's video input as a 3-D array with the count of occurrence of each element (state, action, next state) (s, a, s'), then normalized to have the sum of all elements equals to one. The optimal policy is to always aim at the center of the path.

5.2.4 Driving Rating Calculation

The research compared between TD SARSA Prediction [13], and Reactive SARSA [66]. Reactive SARSA may have an advantage in real-time applications for asynchronous environments. However, after the research evaluated both methods, none

of them is needed in this case. It only needed to calculate the accumulative rewards from the start of driving until the current time. Then normalize it according to the number of frames to get the rating as a percentage. This percentage is different from the RL G variable, as G is the total expected return when following the policy. The equation measures the performance according to what happened, not according to what may happen. The following are the mathematical equations driven and used to calculate the rating of the different driving paths recorded. P_i is noting the path number i. The first frame uses this equation:

$$\text{Rating} \;=\; \text{Reward}_{\,P_i} X \text{ number of Frames}$$

The following frames use:

$$\text{Rating} \;=\; \text{Reward}_{\,P_1} + \frac{\text{Rating} \;\times (\#\text{ of } Fr. - 1)}{\#\text{ of } Fr.}$$

Then calculating the average rating:

$$\text{Rating} \;= \frac{\text{Rating}}{\#\text{ of Fr.}} \times 100\%$$

CHAPTER 6: IMPLEMENTATION AND INITIAL RESULTS OF THE AI CONTROL MODEL

The previous chapters described the implementation of the base ATV architecture to control the ATV using digital signals, and the AI models to drive the ATV. This chapter will go through the AI offline implementation and testing results, the driving performance results, and the real-time design and implementation of the AI model to control the ATV. Performance comparison between this implementation and the implementations discussed in the background chapter will conclude this chapter.

6.1 AI Model Offline Testing and Results

After implementing the LS and LMS models in Chapter 5, the dissertation applied the following six different combinations:

- LS applied to the output of ENet S.S. model
- LS applied to the output of SegNet at low-resolution model
- LS applied to the output of SegNet with high resolution
- LMS applied to the output of ENet S.S. model
- LMS applied to the output of SegNet at low-resolution model
- LMS applied to the output of SegNet with high resolution

The first implementation used the ENet model S.S. on the MacBook Air. The ENet processing speed was 1.1 fps. The performance of the same model using the Nvidia Jetson Nano GPU was 1.4 seconds per frame (0.714 fps) which is too slow for ATV

autonomous driving. As mentioned before, the ENet code is not CUDA-Enabled, making the GPU features of parallel computing useless.

The subsequent trials used the SegNet with the low resolution on the Jetson Nano Nvidia GPU. The SegNet at low resolution was much better with 24 fps, which is acceptable for low-speed autonomous ATV navigation. This fact implies it can detect a new event or change in the path once every 41 ms. Low-speed in this context means an ATV can traverse a path with speed around 16 km/h (10 mph), which is equivalent to 4.4m/s. At this speed, the SegNet can detect a new frame each 18 cm which is acceptable in the forest environment. While in the case of the ENet, it will detect one frame every 6.16 meters, which is not acceptable.

For the SegNet with high resolution on the Jetson Nano GPU, the GPU processed the semantic segmentation at 14 fps with a detection of a new event every 31 cm, assuming the same speed of 10mph, which is acceptable but not preferable.

For better rates, the research set the ATV to move at 6 mph (2.67 m/s), which makes it detect a new frame every 13 cm in case of the low-resolution SegNet model.

The three SS models were examined for the performance once with the LS and once with the LMS. Fig. 6.1 shows the results for the prediction of the LS regression model when using ENet S.S. with the red line is the predicted angle by the ML model and the blue line is the ground truth data gathered from the ATV. The results, in this case, are quite promising as the red line is following the blue line. Note that the blue and red lines are not supposed to be identically matched in this case, as the camera feed was for the ATV while the ATV was driven and following the path, which makes a slight angle toward the center for a correct prediction. However, if the ML model is the ATV driver, the angles should go higher, as if it applies a slight angle, it might go into one of the outer states, leading to more angle value. Fig. 6.2 shows the results for using the LMS model with the ENet SS model. The LMS model shows a weaker ability of the red line to follow the blue line than the LS model in the case of the

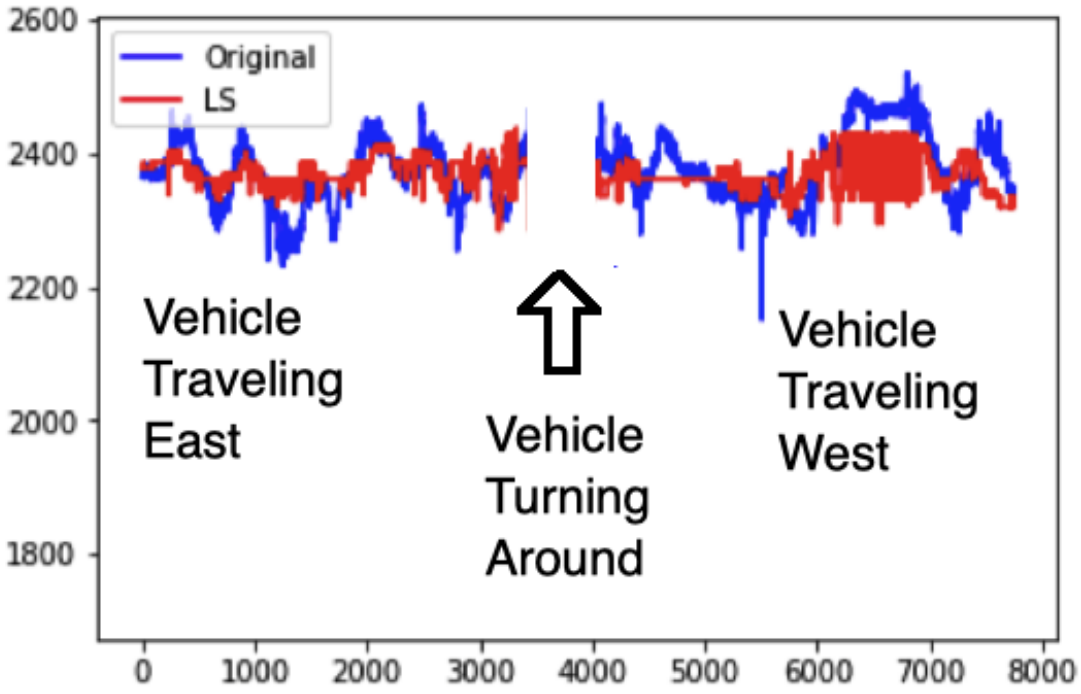

Figure 6.1: Prediction results for ENet S.S. with LS regression model

ENet SS.

Fig. 6.3 shows the results for using the LS model with the SegNet high-resolution SS while Fig. 6.4 shows the results for using the LMS with the SegNet high-resolution model. In the case of the high-resolution SegNet, both LS and LMS models show weak performance in predicting the angles of the steering wheel.

Fig. 6.5 shows the results for using the LS model with the SegNet low-resolution SS. As it appears in the plot, the LS with the SegNet low-resolution model has a good prediction following the ground truth of the steering wheel angle, it may be a little less accurate than the ENet, but it still can be considered as a good result. Fig. 6.6 shows the results for using the LMS with the SegNet low-resolution model. In this case, it appears to have a very weak accuracy in predicting the angle, a bit similar to the SegNet high-resolution performance.

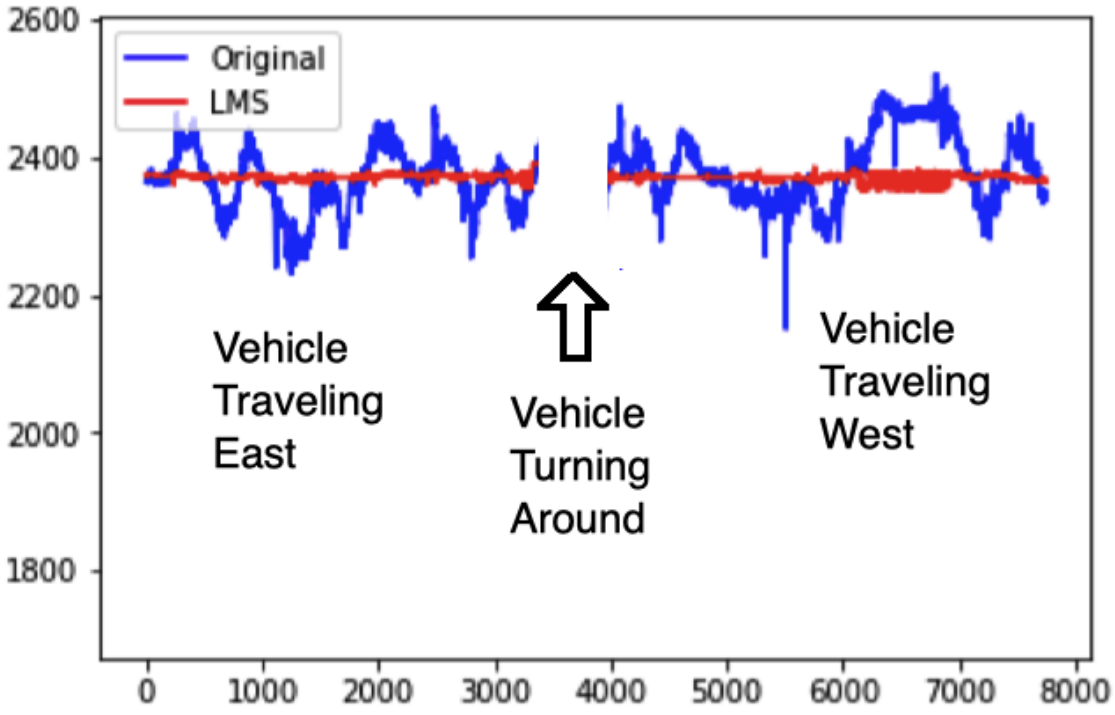

Figure 6.2: Prediction results for ENet S.S. with LMS regression model

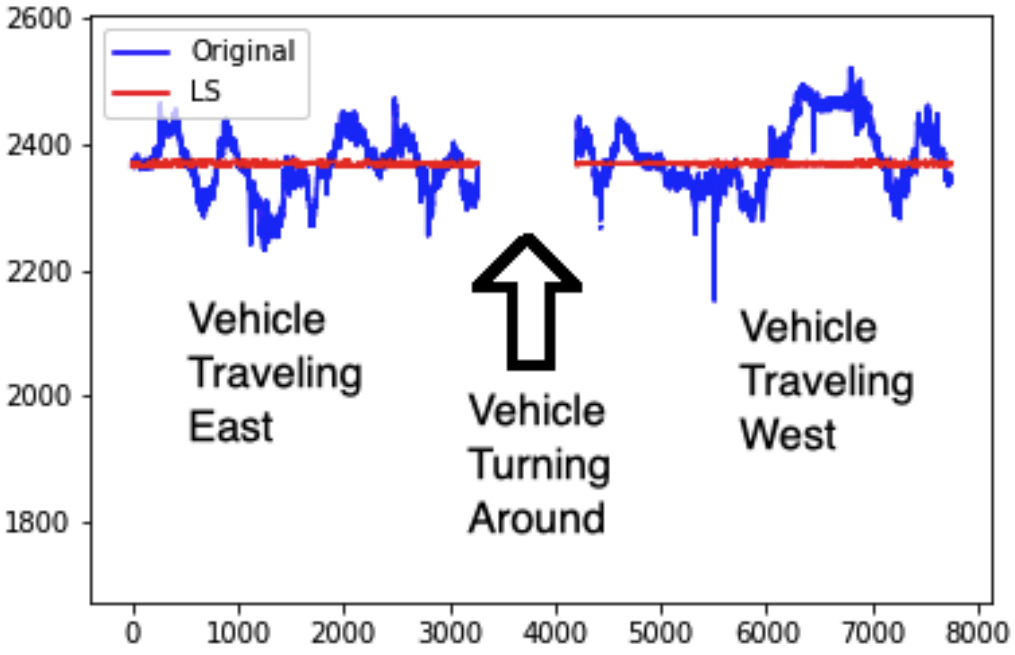

Figure 6.3: Prediction results for SegNet High Resolution S.S. with LS regression model

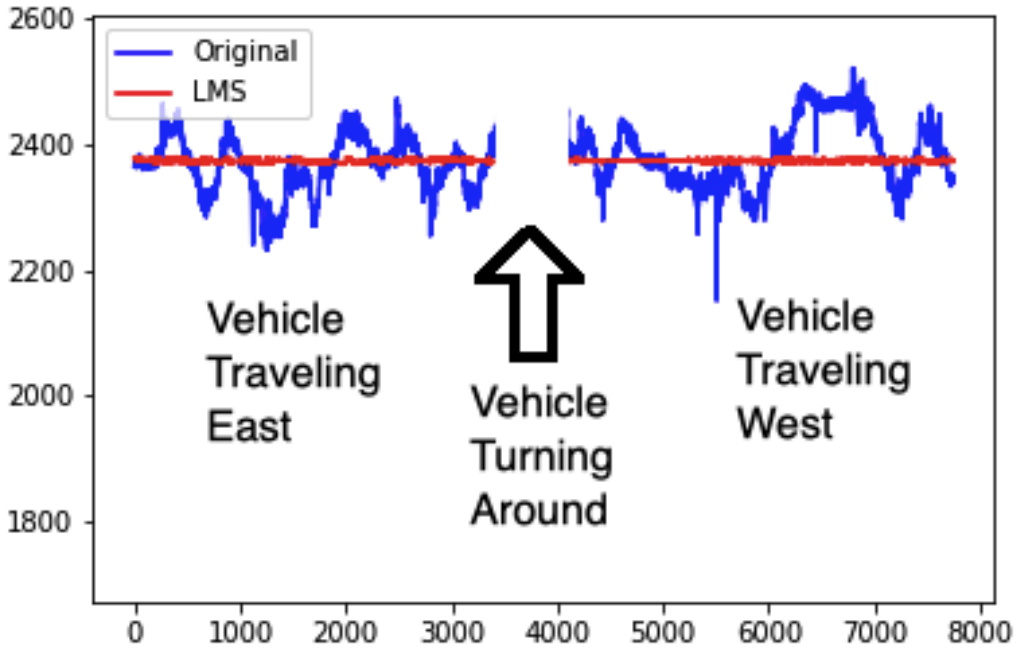

Figure 6.4: Prediction results for SegNet High Resolution S.S. with LMS regression model

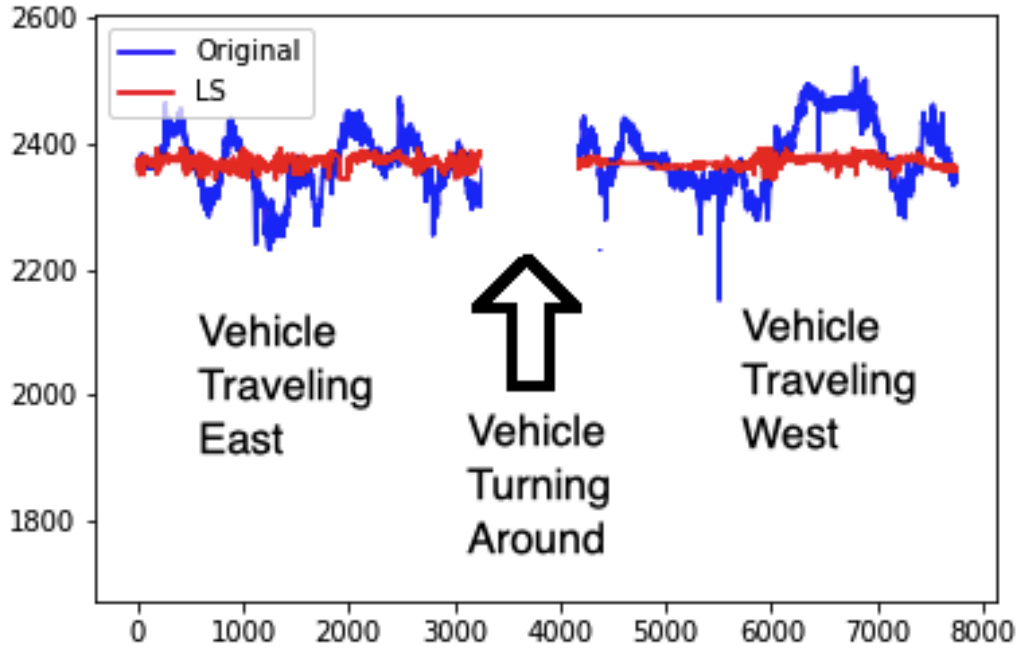

Figure 6.5: Prediction results for SegNet Low Resolution S.S. with LS regression model

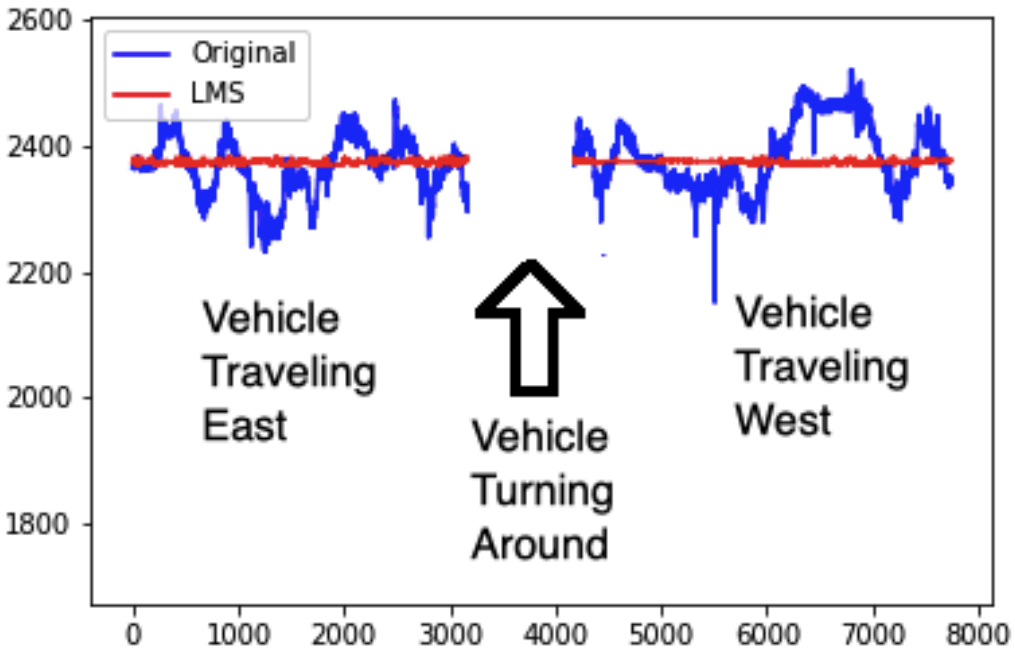

Figure 6.6: Prediction results for SegNet Low Resolution S.S. with LMS regression model

6.2 Testing Results for the AI Driving Performance Measurement Model

This method was applied to the experienced driver to measure his performance, and it appeared 74.3% as shown in Fig. 6.7 on the left.

The researcher applied the model to the seven different driving methods mentioned in Chapter 4, leading to the results in Fig. 6.8. It appears that the zig-zag inside the path is a better driving technique than moving straight outside, as it is always near the center, while the zig-zag outside is the worst as it always gets outside the path. The other driving methods results fluctuate between the zig-zag outside value and the experienced driver.

It appears that the experienced driver might not be efficient to get the rating. However, from the output reward values from the IRL model, it appears that the results were not symmetric around the center. The center was the highest, but the Ro was higher than the Lo. The Lo was a pure zero. These values mean the experienced driver visited the Ro but never the Lo. This proposed explanation could be why

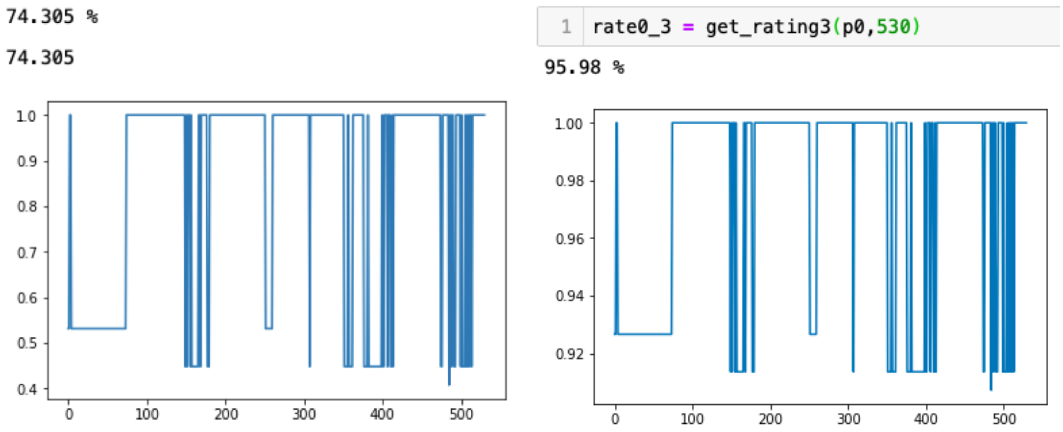

Figure 6.7: First experienced driver:
on the left: First experienced driver evaluation.
on the right: Old experienced driver rated on the average (final) rewards function.

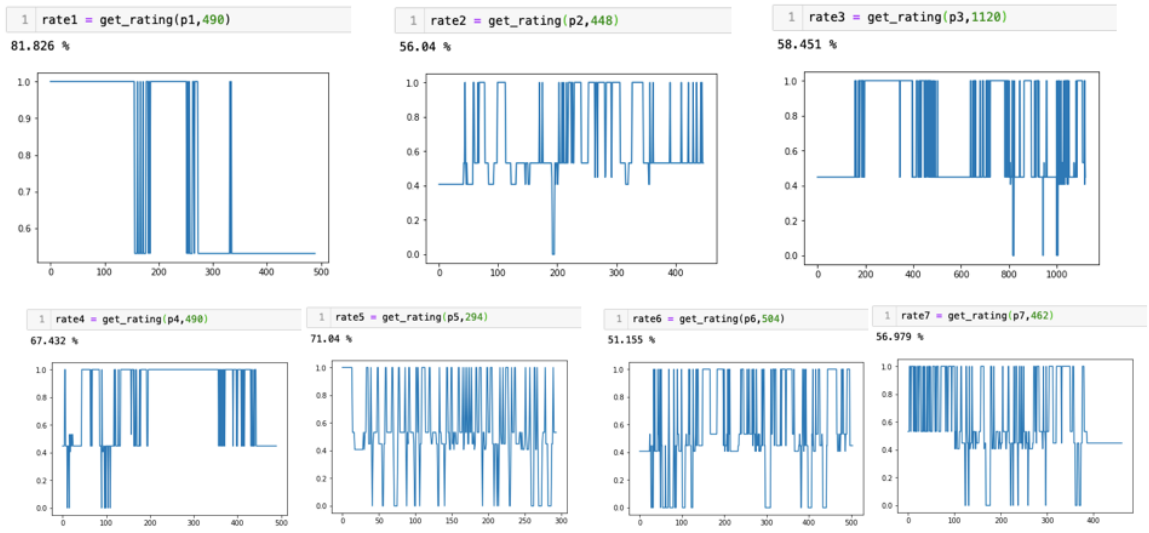

Figure 6.8: Other Drivers Ratings: P1: Ri, P2: Ro, P3: Li, P4: Lo, P5: Zi, P6: Zo, P7: Random

```
In [13]:  1 print (rewards2)

[0.         0.44806205 0.99999994 0.53132512 0.40721582]
```

Figure 6.9: Reward function as output of IRL by first experienced driver - 530 frames

```
In [18]:  1 print(rewards3)

[-3.54757779e-14  9.99999677e-01  9.99963273e-01  9.99999676e-01
  9.99999872e-01]
```

Figure 6.10: Reward function as output of IRL by second experienced driver - 2859 frames

the Ri was higher than the experienced driver average as the right side had a higher value. More experienced driver data may fix this issue. Fig. 6.9 shows the rewards per state visited as the output of the IRL model.

The reward function in Fig. 6.9 shows the highest reward value is when the agent stays in the center. The Ri and the Li have similar reward values. The outside values are much lower. However, the Ro and Lo were not similar, and it shows the experienced driver never visited the left outside state but did visit the right outside.

The researcher performed another data gathering with 2859 frames in addition to the 530 frames used previously. Fig. 6.10 illustrates the reward function. In this case, the rewards are similar for the ro, the ri, the li, and a bit lower for the center, while it was negative for the lo. Applying the model to the new experienced driving attitude using the new reward function and resulted in 99.899% as shown in Fig. 6.11 on the left. This result makes more sense than the first 75% result.

However, to get better results, the two reward functions were weighted and averaged to get the rewards in Fig. 6.12. The new one is much better than the others as the center value is the highest. The Ri and Li had similar values and were higher than the outside states.

The rating for the new experienced driver used the final rewards function (the

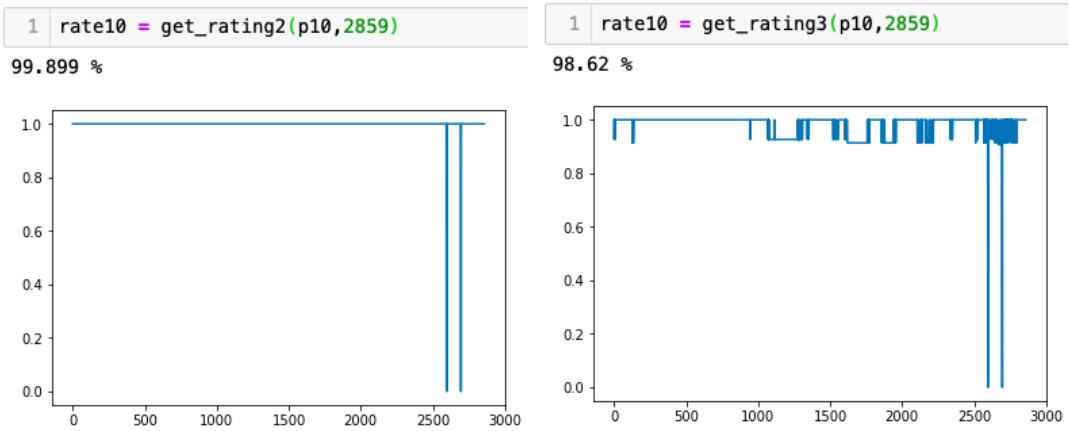

Figure 6.11: Second experienced driver:
on the left: Rating of the second experienced data verses himself.
on the right: New experienced driver rated on the average (final) rewards function.

```
In [36]: 1 rewards_average = ((530 * np.array(r2)) + (rewards3 * 2859))/(530+2859)

In [37]: 1 print(rewards_average)

[-2.99277807e-14  9.13683082e-01  9.99969007e-01  9.26704452e-01
  9.07295373e-01]
```

Figure 6.12: Average rewards for the 530 + 2859 frames

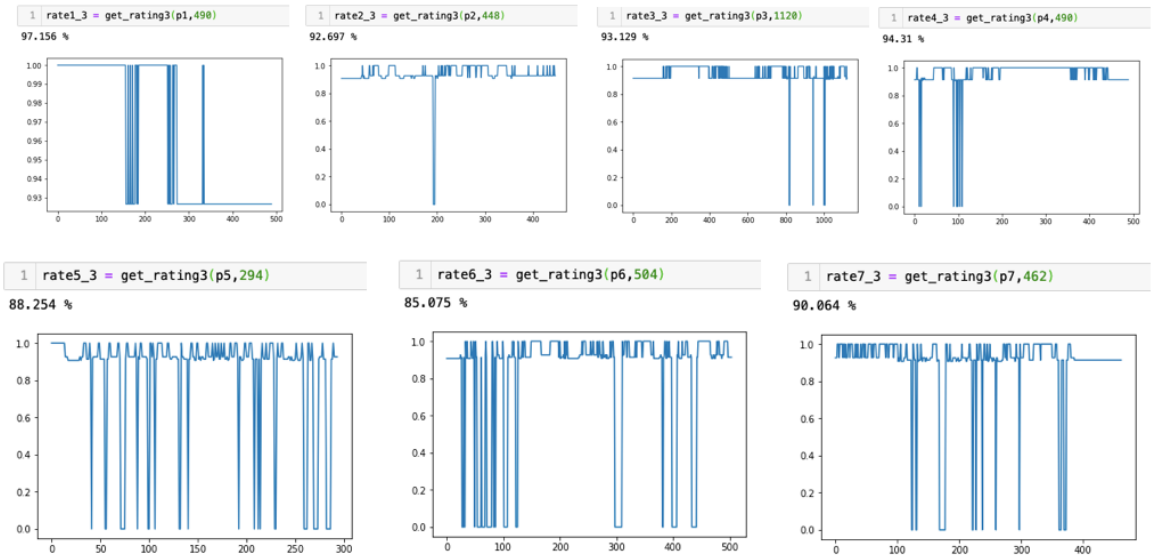

Figure 6.13: Test results for the average (final) rewards function: P1: Ri, P2: Ro, P3: Li, P4: Lo, P5: Zi, P6: Zo, P7: Random

average). The result was 98.62%, as shown in Fig. 6.11 on the right. Fig. 6.7 on the right shows the ranking for the old experienced versus the average rewards function.

Fig. 6.13 shows the results for all other driving techniques. The percentage was higher than the first time. The results now seem more realistic than the first run as the zig-zag in both cases (Zo 85% and Zi 88%) were the lowest results. The random driving technique was the 3rd worst with 90% as it contains all the attitudes together. Followed by the Ro 92%, the Li 93%, the Lo 94%, and the highest was the Ri with 97%. The results show that the new experienced driver is better than all the others, while the Ri is better than the old experienced driver.

6.3 ATV AI Controller Model Real-Time Design and Implementation

After the AI model showed success in predicting the direction of the handlebar in the offline mode, the researcher started to implement the real-time AI model to control the handlebar. This model has a similar implementation as offline, except that the model will work on a frame-by-frame basis instead of operating on the whole video. The GPU used the "Intel Realsense 2" library to read a color frame from

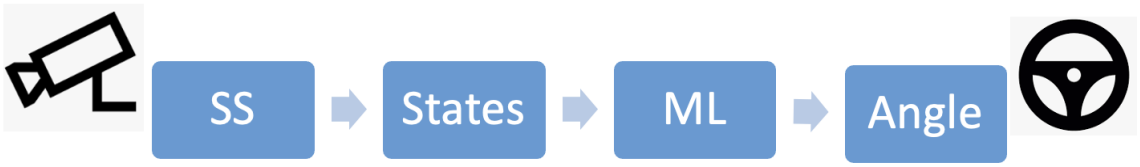

Figure 6.14: The Real-time AI model process to control the ATV handlebar

the D435i camera. The SegNet SS model processes the color frame, and the GPU computes the current and projected positions of the ATV from the SS output. The ML model uses the values of the weights from the training and uses the ATV states to predict the correct angle. The GPU sends the angle and a speed of 2.9 mph using the CAN bus to the ATV actuators. The steering module receives the value and acts accordingly while the speed control system will move the ATV at the appropriate speed. Figure 6.14 illustrates the process for the ATV AI control model.

6.3.1 Handling SS Noise and Path Irregularities

Testing in real-time at first showed some noise on the output of the SS. The cracks and leaves on the paved path may appear as vegetation instead of the paved path. This problem resulted in the ATV trying to avoid what seemed as not path. To fix this problem, the researcher adjusted the image processing portion of the code. Instead of calculating state 1 and state 2 from only one position, the software locates state 1 by scanning the lower 50% of the frame, and the state 2 by scanning the upper window from 50% to 20% of the frame.

6.3.2 Real-Time Testing Results and System Limitations

The researcher tested the ATV in different lightning and weather situations. The results showed the system performs well in sunny weather with shadows on the ground, cloudy weather, rainy weather, fog, and haze. The system performs poorly when tested near sunset (starting an hour before sunset), in very low light conditions, and during heavy rain. The system was not tested in snowing conditions.

The system relies on a camera; therefore, in low light condition, the camera sensor

will not gather enough light and produces a dark image. In this case, the SS will not detect the path contour.

The ATV has a minimum turning radius of 3 meters. On the test path, the sharpest turn has a radius of 10 meters with a curve angle of 20 degrees and the ATV was able to successfully follow the curve. The longest turn was a curved angle of 60 degrees with a radius of 23 meters and the ATV successfully followed the curve.

Video in the reference section demonstrates the ATV moving in the middle of the path most of the time in different locations along the path. This shows the ability of handling turns, going up and down a hill, finding the path is different environments with heavy trees or no trees [67, 68, 69, 70].

6.3.3 Single Run Analysis

For the 2700 frames during test trial, at 2.9 mph speed, in a curvy path with different path elevations and combination of open skies, and between trees [67], the results shows:

- The average position of the ATV is -0.00777 which means very close to the center. The positions are illustrated in Fig. 5.15
- The ATV positions are illustrated in the Table 6.1
- By using the data from the location and apply them into the output of the IRL with weights: 100% for center, 90% for states 1 and -1, 50% for states 2 and -2, and 0% for states 3, -3 and 4, the system exhibits 91.51% of ideal pose as compared to an experienced driving.
- To measure the driving smoothness, the standard deviation of the ATV location was calculated and found to be 1.088 (out of 8 states with mean of -0.00778).
- The time spent on this path was 191 seconds with 9.26 cm per frame. This means the ATV will take a new action after moving 9.26 cm.

Table 6.1: Table illustrating ATV Positions.

ATV State/Position with respect to path	Percentage
4 (path can not be located)	0.00%
3 (outside the path on the right	0.74%
2 (near the path edges but inside the path to the right)	11.52%
1 (slight right)	7.89%
0 (center)	57.78%
-1 (slight left)	10.30%
-2 (near the path edges but inside the path to the left)	11.70%
-3 (outside the path on the left)	0.07%

The upper part of Fig. 6.15 demonstrates the results for the 2700 frames. It shows the current ATV location (state 1) in orange, the projection of the ATV if it continues forward (state 2) in grey, and the steering angle as the output of the ML in blue. To better understand the result, the lower part of Fig 6.15 focuses on only 200 frames, a 14 second section of the pass.

The image on the left represent the ATV on the path at the beginning of the 200 frames. In this picture, the ATV is in the center of the path (state 1 = 0) and the path is going to the left in front of the ATV. Therefore, the ATV will be slightly on the right of the path if it continues straight (state 2 = 1). As a reaction from the ML model, the angle taken is to the left (the blue line is approximately -5 degrees) to keep the ATV in the center of the path.

The graph illustrates that the ML angle is very related to state 2 but in the opposite direction. It appears that the weight of state 2 is bigger than state 1 weight. The graph also shows that the weight contributing in the angle calculation for state 1 is a

positive weight but the state 2 is a negative weight. The weights are driven from the ML model, which reacts to both states together to maintain the ATV on the path. The LR ML model appears here to be beneficial as a trivial thought will be if the current ATV location is on the right, the software should turn to the left to adjust the ATV position. However, the LR ML trained model found that the experienced driver will turn to the left or to the right according to the future trajectory of the path. This means if the ATV is on the right but the path if going to the right, the ATV might turn to the right with a certain angle to maintain the center location of the path and not to go to the left. The LR ML model main purpose was to find the correct equation to control the angle according to state 1 and state 2. Finally, the picture on the right shows the ATV at the end of the 14 seconds section.

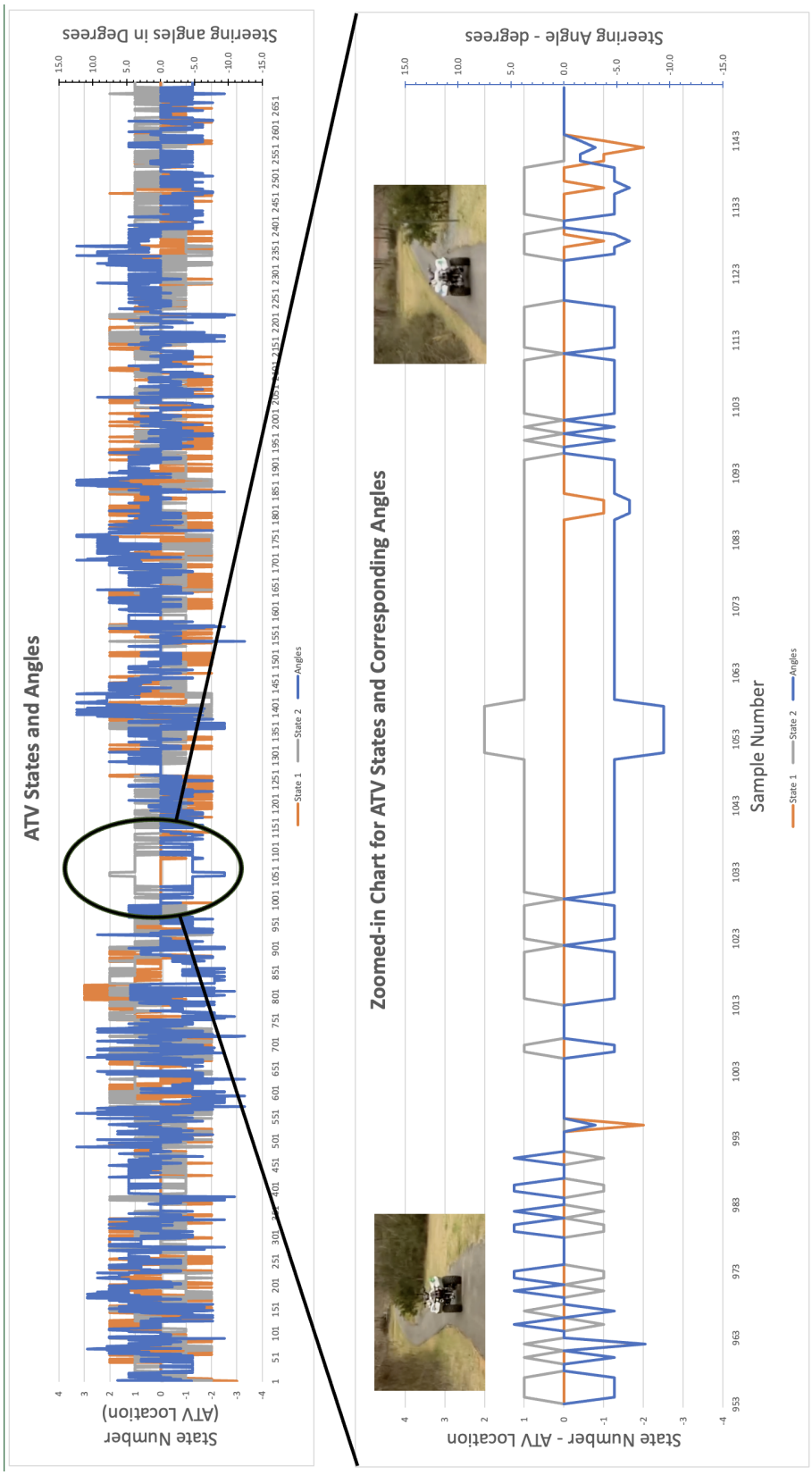

Figure 6.15: Chart from the ATV log file data representing the ATV states (current location and projection) and the angles output of the ML model

CHAPTER 7: CONCLUSION

7.1 Conclusion

An internal combustion engine Honda ATV was transformed into an autonomous ATV that can follow a paved path in a forested environment using embedded systems and AI. The embedded systems is composed of a series of nodes connected via a CAN bus and controls the speed, the steering angle, and the braking system using messages from the AI model. The AI model is implemented on an embedded GPU. The AI model uses the data from a camera with a SS to define the path boundaries, and an image processing model to localize the ATV pose and orientation with respect to the path. A Linear Regression ML model then utilize the pose and orientation to detect the proper steering angle for the ATV to keep following the path. The SS used is a ResNet18 Deep Neural Network and trained using DeepScene dataset. The Linear Regression ML model was trained on a dataset gathered during this research that represents the ATV pose, orientation, and the angle taken by a human driver.

This dissertation demonstrates that it is feasible to control an ATV to follow a paved path in a forested environment using a camera and AI models. The main difference between this research and normal autonomous vehicles is the path detection and vehicle localization. In a normal road, AI models benefit from road signs, lane markings, and all other standardized traffic schemes. This dissertation showed that it is feasible to define the path using a camera and a trained SS model. The main challenge for this work was to position the ATV with respect to the path in different lighting conditions with shadows, leaves, and pavement cracks. The ATV works successfully as long as there is enough light to see which is a trivial condition when using a normal RGB camera. The ATV was able to perform its task in rain, light

fog, cloudy weather, and sunny weather. It currently cannot be operated in heavy fog, heavy rain, or an hour before sunset and later. Offline and online experimental testing took place on the course of this research. The offline testing showed success and promoted the online real-time testing. The results of the real-time testing showed the ATV was capable to move and maintain ideal center pose with 91.5% of the time compared to an experienced driver. The drive showed smooth state transitions with 1.08 standard state deviation from the center of the path out of 8 different states. The research showed the benefit of using linear regression machine learning model to detect the correct angle for the ATV. The researcher applied a constant gain to the ML weights to allow more turning in sharper turns, this gain does not affect the normal turns as the ATV will adjust itself from the camera's feedback.

7.2 Additional Lessons Learned

The ATV was capable of following the path in light foggy weather and in light rain. The ATV functionality was successful as long as the vision range was more than 10 meters. The most important environmental condition for the ATV to perform well is the light. The ATV currently cannot be operated in darkness or low light. Starting one hour before sunset, the ATV capability to detect the path declines rapidly.

An important observation is that the dataset for the ML model was gathered in September and October where the video recorded more green vegetation and trees. Testing occurred at a different environment (at the end of the winter and start of the spring) where the landscape was more grey. Still this condition did not affect the performance of the ATV detecting its pose and orientation with respect to the paved path.

Another observation was that the ML model put more weight on state 2 (ATV projection position with respect to the path) and less on state 1 (current ATV location with respect to the path). This showed the importance of state 2 as the ML model will react more to state 2 and less to state 1. Another advantage from using the

ML model training was to get the proper weights and the constant bias for the ATV steering angle formula depending on the current pose and pose projection.

Since the dataset is composed of only 3 variables (inputs: state 1, state 2, and output: the steering angle), a regression model was needed to calculate the value of the angle. A linear regression model showed good results for angles calculation to maintain the ATV on the paved path.

During testing, it was recommended to test the ML results after training the model using more data. The results did not improve much after increasing the dataset size from 33K entries to 128K entries.

For the ML to control the ATV steering angle, it was important to test the ATV on lower speeds in order for the researcher to have more control over the ATV in emergency situations and to allow the ML model more time to make a proper decision. As mentioned before, the ATV engine is classified as an internal combustion engine. It was noticed that controlling the speed on such an engine is a challenge. It would be much easier if the ATV was an electric vehicle with electric motors. This would allow more time to focus on the AI model instead of controlling the ATV speed. The research showed a successful speed control of 2.9 mph and above (though, as shown above, the vehicle operated at the low end of this range).

The steering control showed that the external motor was more powerful and able to reach better accuracy from just relying on the vehicle power steering-assist motor. The new implementation showed better accuracy reaching a specific angle than the previous implementation mostly because of the caster angle. Also the tension of the cable and any associated slack did not affect the performance of the steering because of the feedback loop from the steering axis.

The SS model was trained using a similar but not the same environment (DeepScene dataset) which is a forest environment in Germany. The results of the SS model were not always as desired depending on the weather situation and the pavement

cracks and otherwise. These results had some noise that needed more effort from the image processing part of the model to remove and detect the correct ATV pose and orientation.

The ATV was able to follow a gravel path as well, not only asphalt. This appeared in one of the testing results. The main reason is that the SS model is trained to classify everything as either trees, or vegetation, or paved path.

7.3 Future Work

For the future work, it is recommended that, since the world is moving toward electrical vehicles, to perform the same tests on an electrical ATV.

More testing would add better understanding for the system limitation like testing the ATV in the snow.

This dissertation focuses on the ATV following a single path in a forested environment. It does not handle a fork and making a choice. As a future investigation, using the ML to choose which path to take by integration of this study with the breadcrumb sensors [47, 71], a parallel study that had place in the same lab as this dissertation.

A good improvement is to use a more powerful GPU that can process the AI model and gather more data at the same time.

Fine tuning the SS using local data might help eliminate the noise and reduce the processing power needed by the image processing portion of the model. Besides, increasing the number of classes to include other potential subjects and train the ML accordingly may enhance the ATV performance to include different environment with less limitations.

.

REFERENCES

[1] "Electric Cars, Solar Panels & Clean Energy Storage | Tesla." https://www.tesla.com/.

[2] K. H. Erian, "System integration over a can bus for a self-controlled, low-cost autonomous all-terrain vehicle," Master's thesis, The University of North Carolina at Charlotte, 2019.

[3] "Valeo - Smart technology for smarter cars." https://www.valeo.com/en/.

[4] "Future of mobility overview | deloitte insights." https:// www2.deloitte.com/ insights/us/ en/focus/ future-of-mobility/ overview.html? id=us:2ps:3gl: confidence:eng:cons:102218: nonem: na: zsXNn12x: 1123801355:339149905178 :e: Future_of_Mobility: Future_of_Mobility_Exact:nb.

[5] A. Valada, G. Oliveira, T. Brox, and W. Burgard, "Deep multispectral semantic scene understanding of forested environments using multimodal fusion," in *International Symposium on Experimental Robotics (ISER)*, 2016.

[6] K. H. Erian, S. Mhapankar, S. Gambill, and J. M. Conrad, "System Integration over a CAN Bus for a Self-Controlled, Low-Cost Autonomous All-terrain Vehicle," in *IEEE SoutheastCon 2019*, pp. 1–8, IEEE, 2019.

[7] B. B. Rhoades and J. M. Conrad, "A survey of alternate methods and implementations of an intelligent transportation system," in *IEEE SoutheastCon 2017*, pp. 1–8, IEEE, 2017.

[8] B. B. Rhoades, D. Srivastava, and J. M. Conrad, "Design and Development of a ROS Enabled CAN Based All-Terrain Vehicle Platform," in *IEEE SoutheastCon 2018*, pp. 1–6, IEEE, 2018.

[9] B. B. Rhoades and J. M. Conrad, "A Novel Terrain Topology Classification and Navigation for an Autonomous CAN Based All-Terrain Vehicle," in *IEEE SoutheastCon 2018*, pp. 1–6, 2018.

[10] K. H. Erian, P. H. Regalado, and J. M. Conrad, "Missing data handling for machine learning models," in *IAES International Journal of Robotics and Automation (IJRA) 2021*, pp. 123–132, 2021.

[11] K. H. Erian and J. M. Conrad, "Controlling atv steering angle on a forest path using machine learning," in *IEEE SoutheastCon 2021*, pp. 1–7, 2021.

[12] "A 2019 Guide to Semantic Segmentation | Derrick Mwiti." https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation -ca8242f5a7fc.

[13] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge, 1998.

[14] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *International Conference on Machine Learning (ICML)*, vol. 1, 2000.

[15] M. Alger, "Inverse reinforcement learning." Zenodo 2016. https://doi.org/10.5281/zenodo.555999, 2016.

[16] B. Eysenbach, J. Tyo, S. Gu, G. Brain, R. Salakhutdinov, Z. Lipton, and S. Levine, "Reinforcement learning with unknown reward functions," in *Task-Agnostic Reinforcement Learning Workshop at ICLR 2019*, International Conference on Learning Representations (ICLR), 2019.

[17] "End-to-End Deep Reinforcement Learning without Reward Engineering â The Berkeley Artificial Intelligence Research Blog," 2019.

[18] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, "Active Reward Learning," in *Robotics: Science and Systems*, 2014.

[19] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid Reward Architecture for Reinforcement Learning," in *Advances in Neural Information Processing Systems*, pp. 5392–5402, 2017.

[20] J. Clark and D. Amodei, "Faulty reward functions in the wild," *Internet: https://blog.openai.com/faulty-reward-functions*, 2016.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings, Twenty-First International Conference on Machine Learning (ICML) 2004*, pp. 1–8, 2004.

[22] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[23] D. Ramachandran and E. Amir, "Bayesian Inverse Reinforcement Learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 7, pp. 2586–2591, 2007.

[24] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.

[25] S. Levine, Z. P. PopoviÂŽc, and V. Koltun, "Nonlinear Inverse Reinforcement Learning with Gaussian Processes," in *Advances in Neural Information Processing Systems*, pp. 19–27, 2011.

[26] K. Brookhuis, D. De Waard, and B. Mulder, "Measuring driving performance by car-following in traffic," *Ergonomics*, vol. 37, no. 3, pp. 427–434, 1994.

[27] M. R. Savino, "Standardized names and definitions for driving performance measures," Master's thesis, Tufts University, 2009.

[28] C. You, J. Lu, D. Filev, and P. Tsiotras, "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning," *Robotics and Autonomous Systems*, vol. 114, pp. 1–18, 2019.

[29] J. Ferreira, E. Carvalho, B. V. Ferreira, C. de Souza, Y. Suhara, A. Pentland, and G. Pessin, "Driver behavior profiling: An investigation with different smartphone sensors and machine learning," *The peer-reviewed open access scientific journal published by the Public Library of Science(PLOS ONE)*, vol. 12, p. e0174959, Apr 2017.

[30] M. Shimosaka, T. Kaneko, and K. Nishi, "Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning," in *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pp. 1694–1700, Nov 2014.

[31] M. Shimosaka, K. Nishi, J. Sato, and H. Kataoka, "Predicting driving behavior using inverse reinforcement learning with multiple reward functions towards environmental diversity," in *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 567–572, Aug 2015.

[32] A. Gupta, P. S. Merchant, *et al.*, "Automated lane detection by k-means clustering: a machine learning approach," *Electronic Imaging*, vol. 2016, no. 14, pp. 1–6, 2016.

[33] F. Beainy and S. Commuri, "Development of an autonomous atv for real-life surveillance operations," in *2009 17th Mediterranean Conference on Control and Automation*, pp. 904–909, 2009.

[34] S. Lee, S. Kim, and B. Song, "A complete fault detection and diagnosis system for autonomous all-terrain vehicle (atv)," in *Proceedings of the 11th IASTED International Conference*, vol. 633.

[35] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[36] N. Sanil, P. A. N. venkat, V. Rakesh, R. Mallapur, and M. R. Ahmed, "Deep learning techniques for obstacle detection and avoidance in driverless cars," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, pp. 1–4, 2020.

[37] R. R. O. Al-Nima, T. Han, and T. Chen, "Road tracking using deep reinforcement learning for self-driving car applications," in *International Conference on Computer Recognition Systems*, pp. 106–116, Springer, 2019.

[38] Z. Liu, Y. Cai, H. Wang, L. Chen, H. Gao, Y. Jia, and Y. Li, "Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2021.

[39] M. Hirz and B. Walzel, "Sensor and object recognition technologies for self-driving cars," *Computer-aided Design and Applications*, vol. 15, no. 4, pp. 501–508, 2018.

[40] S. S. Rao and S. R. Desai, "Machine learning based traffic light detection and ir sensor based proximity sensing for autonomous cars," *Proceedings of the International Conference on IoT Based Control Networks Intelligent Systems - ICICNIS 2021*, 2021.

[41] H.-Y. Chou, F. Khorsandi, and S. G. Vougioukas, "Developing and testing a gps-based steering control system for an autonomous all-terrain vehicle," in *2020 ASABE Annual International Virtual Meeting*, p. 1, American Society of Agricultural and Biological Engineers, 2020.

[42] H. Mousazadeh, "A technical review on navigation systems of agricultural autonomous off-road vehicles," *Journal of Terramechanics*, vol. 50, no. 3, pp. 211–232, 2013.

[43] J. AKSHAY, "Modification on An Existing ATV Steering to Supplement its Autonomous Functionality," 2020.

[44] R. A. McKinney, M. J. Zapata, J. M. Conrad, T. W. Meiswinkel, and S. Ahuja, "Components of an autonomous all-terrain vehicle," in *IEEE SoutheastCon 2010*, pp. 416–419, IEEE, 2010.

[45] A. Cortner, J. M. Conrad, and N. A. BouSaba, "Autonomous all-terrain vehicle steering," in *IEEE SoutheastCon 2012*, pp. 1–5, IEEE, 2012.

[46] J. R. Henderson, J. M. Conrad, and C. Pavlich, "Using a CAN bus for control of an All-terrain Vehicle," in *IEEE SoutheastCon 2014*, pp. 1–5, IEEE, 2014.

[47] D. P. Grabowsky, J. M. Conrad, and A. F. Browne, "A breadcrumb system for assisting outdoor autonomous robots with path identification and localization," in *IEEE SoutheastCon 2021*, pp. 1–6, IEEE, 2021.

[48] R. Cuffley, "Amp Tyco Superseal waterproof connector assembly guide for wiring boats cars, well anything - YouTube." https://www.youtube.com/watch?v=OgrA_6zFqqY.

[49] S. K. Gurram and J. M. Conrad, "Implementation of CAN bus in an autonomous all-terrain vehicle," in *IEEE SoutheastCon 2011*, pp. 250–254, IEEE, 2011.

[50] "MSP430 CAN interface - electrodummies." http://www.electrodummies.net/en/msp430-2/msp430-can-interface/.

[51] "Compiler/diagnostic messages/MSP430/1528 - Texas Instruments Wiki." http://processors.wiki.ti.com/index.php/Compiler/diagnostic_messages/MSP430/1528.

[52] "MSP430 SPI Peripheral | Argenox." https://www.argenox.com/ library/ msp430/ msp430-spi-peripheral-chapter-9/.

[53] "CAN Bus Shield." https://github.com/Seeed-Studio/CAN_BUS_Shield/blob/ 54510776ffafab1074693dc2884335f995a24242/mcp_can.cpp.

[54] "MCP2515 NOTES." http://ww1.microchip.com/ downloads/ en/ devicedoc/ 20001801h.pdf.

[55] "Build an Arduino CAN Bus Network | Henry's Bench." http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/arduino-can-bus-module-1st-network-tutorial/.

[56] "Steering Theory Chapter 7: Steering Dynamics." http://ckw.phys.ncku.edu.tw/ public/ pub/ Notes/ GeneralPhysics/ Powerpoint/ Extra/ 05/ 11_0_0_Steering_Theroy.pdf.

[57] "Learn About Positive and Negative Camber, Caster, and Toe." https:// www.comeanddriveit.com/ suspension/ camber-caster-toe.

[58] M. Zhuk, V. Kovalyshyn, and R. Tcir, "Defining duration of driver reaction time components using the neurocom complex," *ECONTECHMOD: An International Quarterly Journal on Economics of Technology and Modelling Processes*, vol. 4, 2015.

[59] "Omni calculator." https://www.omnicalculator.com/physics/stopping-distance, 2021.

[60] K. H. Erian and J. M. Conrad, "Measuring driving performance for an all-terrain vehicle in a paved road in the woods," in *IEEE SoutheastCon 2020*, pp. 1–8, IEEE, 2020.

[61] "Canny Edge Detection using Matlab | Rachmawan(2019)." https://www.mathworks.com/matlabcentral/fileexchange/46859-canny-edge-detection.

[62] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] "Semantic segmentation with OpenCV and deep learning | Adrian Rosebrock." https://www.mathworks.com/matlabcentral/fileexchange/46859-canny-edge-detection.

[64] D. Franklin, "Semantic Segmentation with SegNet." https://github.com/dusty-nv/jetson-inference/blob/master/docs/segnet-console-2.md.

[65] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[66] J. B. Travnik, K. W. Mathewson, R. S. Sutton, and P. M. Pilarski, "Reactive Reinforcement Learning in Asynchronous Environments," *Frontiers in Robotics and AI*, vol. 5, p. 79, 2018.

[67] K. Erian, "My PhD Dissertation: Autonomous ATV following a green path at UNC Charlotte using AI models - YouTube." https://youtu.be/H4YbS3nC0cI, 2022.

[68] K. Erian, "My Ph.D. Dissert.: Autonomous ATV following a green path at UNC Charlotte using AI from Phillips Rd - YouTube." https://youtu.be/kIbjwfwX928, 2022.

[69] K. Erian, "My Ph.D. Dissert.: Autonomous ATV following green toward EPIC by K. ERIAN - YouTube." https://youtu.be/tavz4JUDM_g, 2022.

[70] K. Erian, "ERIAN Autonomous ATV following a pebble path in a forest environment - YouTube." https://youtu.be/3VhO5c2PBlU, 2022.

[71] D. P. Grabowsky, S. Shue, and J. M. Conrad, "Machine learning for wireless distance estimation model parameter estimation for breadcrumb localization applications," in *IEEE SoutheastCon 2022*, pp. 1–8, IEEE, 2022.