Relative Position Vector Generation with Computer Vision for Vehicle Platooning Applications

by

Tyler Flegel

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 5th, 2023

Keywords: Relative Position Vector, RPV, Path Following, Camera, Computer Vision, Monocular, Stereoscopic

Approved by

David Bevly - Bill and Lana McNair Distinguished Professor of Mechanical Engineering
Howard Chen - Assistant Professor of Industrial and Systems Engineering
Chad Rose - Assistant Professor of Mechanical Engineering

Abstract

Many significant advances have been made in autonomous vehicle technology over the recent decades. This includes platooning of heavy trucks. As such, many institutions have created their own version of the basic platooning platform. This includes the California PATH program [1], Japan's "Energy ITS" project [2], and Auburn University's CACC Platform [3]. One thing these platforms have in common is a strong dependence on GPS based localization solutions. Issues arise when the platoon navigates into challenging environments, including rural areas with foliage which might block receptions, or more populated areas which might present urban canyon effects. Recent research focus has shifted to handling these situations through the use of alternative sensors, including cameras.

The perception method proposed in this thesis utilizes the You Only Look Once (YOLO) real-time object detection algorithm in order to bound the lead vehicle using both RGB and IR cameras. Two different YOLO variants were evaluated: YOLOv3 and TinyYOLOv3. Monocular range is determined using both the classical pinhole model and virtual horizon ranging model. A bearing model is introduced which uses the range to determine bearing to the lead vehicle. Various combinations of cameras, YOLO models, and ranging models are then tested on heavy duty truck data collected on roads near Auburn University's NCAT facility. The results shown in this thesis reveal that there is a slight range accuracy advantage to YOLOv3 on both the pinhole camera model and the Virtual horizon model. TinyYolo was shown to have a faster processing speed which would be ideal in highly dynamic situations.

Using the results from the on road truck analysis, a real-time implementation was developed using two consumer sedans. During analysis it was discovered that YOLO had momentary lapses in which it would not detect the lead vehicle, and would therefore not be able to provide range and bearing measurements. To address this, a sub-tracking algorithm

was developed. The algorithm was developed around established tracking algorithms, and analysis was performed to determine which tracking algorithm was best suited for dynamic vehicle tracking. Additionally, a slight variation of the method was developed which utilized a stereoscopic camera. The sub-tracking algorithm and stereoscopic vehicle detection algorithm were evaluated in several real-time platooning scenarios, in which the following vehicle operated autonomously using lateral control.

Acknowledgments

Throughout the process of completing my masters I endured many challenges, both personal and professional, which would not have been possible to overcome without the help of those listed here.

I would first like to thank Dr. Bevly for the opportunity to work in the GAVLAB and further my education. I would also like to thank Howard Chen for sharing his knowledge and love of robotics, and allowing me to develop the project for his class. Also thanks for the free ping pong lessons.

My accomplishment would not have been possible without the support of my fellow lab members. I thank Kathleen Steadman for her constant support and willingness to proofread, and also for distracting me with the daily crossword puzzle. Thanks to John David "The Tech Priest" Sprunger, for sharing his extensive knowledge and expertise, even in the middle of the night. I thank Greg Mifflin for his help on many projects and his openness to being a sounding board for all of my off the wall ideas. He has been a good friend, and I look forward to continuing to work with him.

I would like to thank all of my teachers from Lincoln High School for their passion and effort. They provided me with the framework to continue my education. I would especially like to thank Brian Kelly for sparking an interest in engineering and technology, an interest which continues to this day.

I would also like to thank my family for their love and support. I thank my mom, Rita, for always being available when times were tough. I would also like to thank my dad, Mike, for sharing his ideas and insight which was always useful. I would like to thank Chris Deaton for his engineering knowledge, he is one of the best engineers I have ever had the pleasure of working with, and I aspire to be more like him. Also thanks to my grandparents, Don and

Ilse Harcrow, for their love and support, the lessons they taught me are useful on a daily basis.

I would like to thank Mac, Lilly, Cheese, and Romano Jones. Without their unconditional love and unwavering support I would have become mentally unstable a long time ago, and none of this would have been possible. I love them all dearly.

Finally, I would like to acknowledge my girlfriend Becky Jones. These past 3.5 years have been amazing. She is responsible for so much of my personal growth and I am a better person because of her. I cannot imagine accomplishing any of this without her and I cannot express here how much she means to me. I love her so much and I cannot wait for our future life together.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

Platooning can be defined as a convoy of 2 or more vehicles, in which the first is manually driven by an operator. There are many advantages to such a setup. One advantage is a reduction in the number of drivers required to operate the platoon. Another advantage is an increase in the fuel efficiency of all vehicles in the platoon, including the lead vehicle [4]. In order to perform platooning maneuvers, a Relative Position Vector (RPV) containing range and bearing to a lead vehicle, is traditionally used. Platooning platforms often have a strong dependence on GPS based solutions to generate the RPV, such as Dynamic Real Time Kinematic (DRTK) to provide localization with centimeter level accuracy [5]. Due to this dependence, issues arise when platoons enter areas that do not have clear GPS signals. Platoons also face issues with multipath errors, which occur when GPS signals are reflected off of surfaces and cause invalid GPS solutions. Real Time Kinematic (RTK) based solutions are especially vulnerable to multipath errors [6]. Additionally, RTK-based solutions fail when even a singular truck in the platoon loses GPS reception. A visual representation of how vegetation can limit GPS satellite reception to otherwise viable satellites can be seen in Figure 1.1. These limitations associated with GPS have shifted research focus to find solutions that do not rely on a GPS solution for platoons to function properly.

Sensors including LiDAR, Radar as well as monocular and stereoscopic cameras can be used as alternatives to GPS. LiDAR is effective at generating point clouds which can be used for identifying and determining range to a lead vehicle; however, LiDAR is typically more expensive than the other options. Radar is commonly used for Adaptive Cruise Control in both commercial and privately owned vehicles. However, the methods employed simply

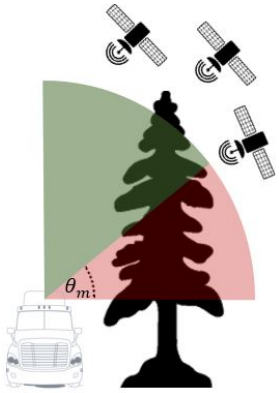

Figure 1.1: Depiction of blocked satellites which would otherwise be viable for a GPS solution.

detect range and range rate to points in the surrounding area, and algorithms must contextualize the points. This would be required for sophisticated path following, such as for a lane change. Traditionally, for vision problems requiring range measurements, stereo cameras are typically chosen. However, stereo cameras have some drawbacks. One issue is that they are more expensive than monocular cameras, in general. Additionally, stereo cameras require the use of an algorithm, which solves the correspondence problem, to generate depth maps. When tracking objects that are far away, relative to the distance between the two cameras, the method can fail [7]. The algorithm can also fail when dealing with objects and environments that do not have enough detectable features. Monocular cameras are inexpensive, and readily available. RGB cameras are the most popular monocular camera, and capture visible light. IR monocular cameras record infrared light instead of visible light. This makes it possible for them to function at night. They are also more resilient to dust and weather effects than their RGB counterparts. The purpose of this thesis is to develop and evaluate methods for estimating the RPV to a leader vehicle in platoon through the use of a monocular camera (IR and RGB) on real world data. Additionally, the methods will be explored in real time on smaller consumer vehicles with both monocular and stereoscopic cameras.

1.2 Related Work

1.2.1 Scale Platooning Platforms

Due to the costs and logistics associated with having and maintaining a full scale platooning platform, much of the research on platooning has been done with scale platooning platforms. This extends to scale platooning with monocular camera. Rezgui et al. developed a platform that used Tiny YOLO (You Only Look Once: Unified, Real-Time Object Detection [8]) to bound the lead vehicle, as seen in Figure 1.2, and determine range from ultrasonic sensors [9]. The robot's wheel speed commands were determined based on the lead vehicle's x-coordinate in the image frame.

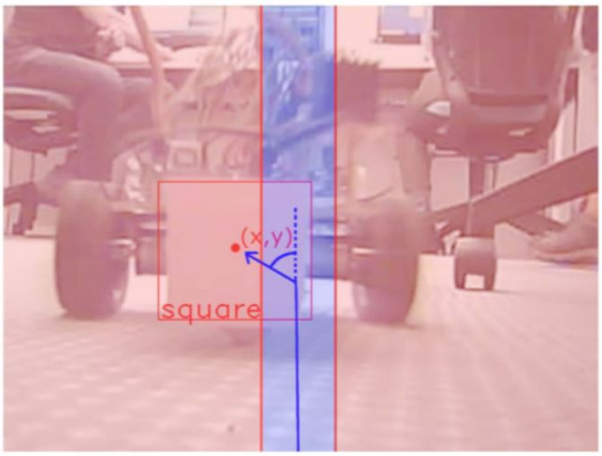

Figure 1.2: Scale Platooning Platform [9].

A similar scale platooning platform was developed in [10], that instead used a Deep Neural Network (DNN) to bound the lead vehicle. The size of the bounding box was used to control the speed of the follower vehicle. The steering angle was determined in a similar fashion to [9]. Another example of a scale platooning platform is [11] which used a classical HSV (Hue Saturation Value) filter to bound the lead vehicle. The range was determined using a derived function from the area of this bounding box. The inter-vehicular angle was determined using the lateral offset and range. Theinert et al. [12] used a similar method, except mounted an illuminated sphere to the rear of the lead vehicle which was detected using classical methods.

Additionally, the pinhole camera model was used to get range. Mitchell et al. detected a mock license plate using a Pixy Camera to get range [13]. Another use for the YOLO framework was in [14], which not only detected the lead vehicle, but used an LED light matrix to communicate specific vehicle states to the follower.

1.2.2 Full-Scale Platooning Platforms

While most of the work on monocular based platooning has been on scale platforms, it has been extended to full-scale platforms. Researchers in [15] used YOLOv3 in tandem with a dedicated keypoint localization network to estimate lead vehicle states through the use of a Kalman filter and other sensors on pre-recorded data. Kim et al. used YOLO to detect the lead vehicle, as well as other vehicles on the road way in order to detect cut-ins [16]. Example algorithm detections of the lead vehicle and other road vehicles are shown in Figure 1.3. In the methodology, RADAR was used to determine range, and this range was used to generate a platooning-specific lane following algorithm.

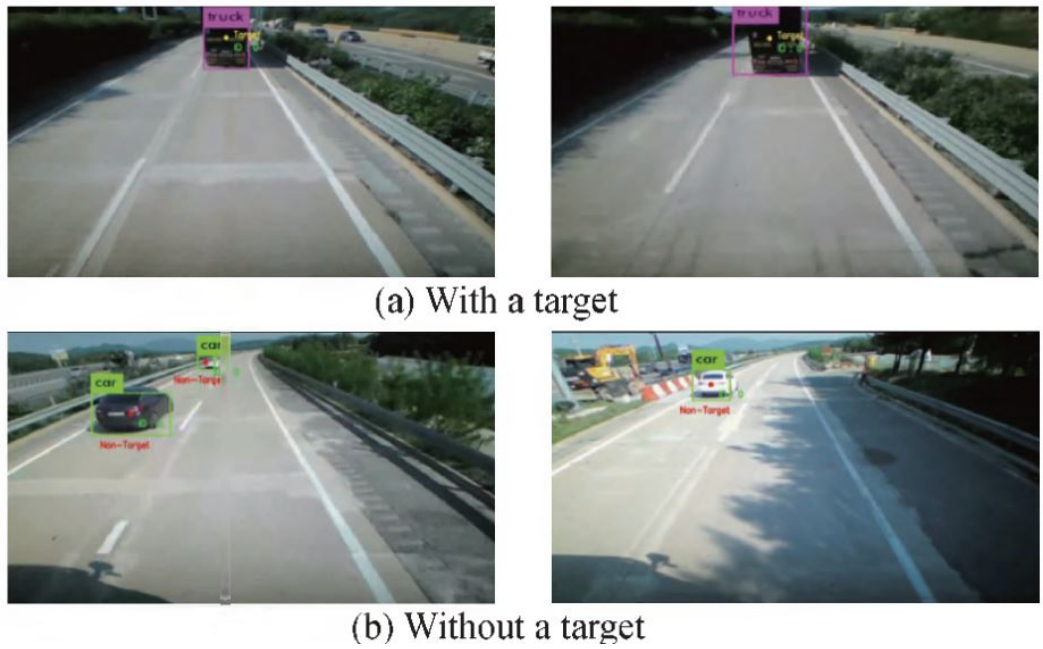

Figure 1.3: YOLO implementation from [16] which detected the lead vehicle and other surrounding vehicles to help contextualize radar points.

Some work has been done on monocular infrared platooning. Woeber et al. evaluated the accuracy of the AdaBoost detection algorithm on thermal imagery of heavy trucks, but did not determine range or bearing to the vehicle [17]. A popular approach to the monocular platooning problem involves using a predefined 3D model or template of the lead vehicle, defining features on the model, and then finding these features in real time to determine pose using classical computer vision [18-21]. Figure 1.4 shows the the features detected and used to align the camera data to the predefined model.

Figure 1.4: Relative Pose Determined using a Camera and Predefined Model [19].

Pierre et al. applied a Correction-Based Incremental Learning (CBIL) approach to the vision based platooning problem [22]. The concept involved using a traditional DNN to associate the appearance of the lead vehicle with specific driving commands in a simulated environment. The DNN was trained offline with limited training data, but the proposed CBIL approach was utilized to procure unique new training data at instances when the DNN made an inaccurate prediction during operation. This new training data was aggregated with existing training data, and used to train an improved DNN. The CBIL algorithm was run online during platooning scenarios, and the training was done offline with the new dataset. The results were compared against a traditional supervised learning approach which had a larger dataset. The CBIL algorithm resulted in a better performance than the supervised learning approach.

1.2.3 Other YOLO Ranging Applications

Outside of platooning applications, YOLO has been used to determine monocular range. A collision warning system used YOLO outputs as the input to a secondary neural network to determine range to pedestrians and street vehicles for locomotives. A secondary neural network was trained to determine the range to the detected entities [23]. YOLO has also been utilized to detect street signs [24]. The ranges to the signs were determined using the size of the bounding box. The method applied to a stop sign can be seen in Figure 1.5 along with its accompanying bounding box which is used for the ranging method.

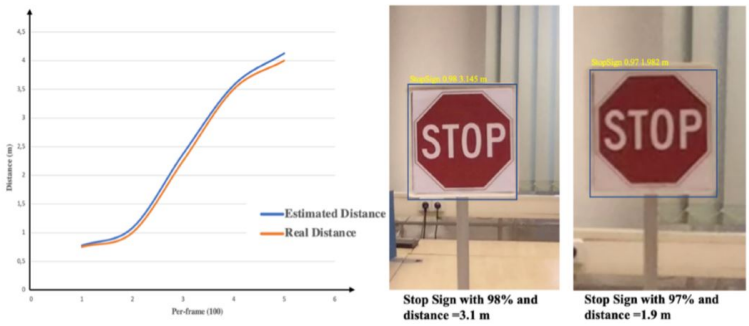

Figure 1.5: YOLO implementation trained to detect and determine range to common street signs [24].

1.2.4 Monocular Ranging

Monocular cameras have been utilized for Adaptive Cruise Control (ACC). Researchers in utilized cameras for determining range and range rate [31]. Monocular cameras have also been utilized as part of a forward collision warning system [26]. The method presented does not rely on the pinhole camera model, instead estimating a virtual horizon line to determine range to oncoming vehicles. This forward collision warning system perception method was extended in [27], which utilized lane lines to avoid estimating vehicle width.

1.3 Contributions

The contributions of this thesis can be broken into two major categories. The first, as was presented at GVSETS 2022, involves determining the effectiveness of an open source object detection algorithm on platooning data in order to guide future development [64]. The second involves taking this information and applying it to a platform in real-time. The exact contributions are listed below.

- Evaluation of open source object detection algorithm based range and bearing measurements on real world data in varied road conditions with varied cameras.
  - RGB Camera vs. IR Camera
  - Pinhole Camera Model vs. Virtual Horizon Model
  - Dirt vs. Paved Roads
  - Full YOLO vs. Tiny-YOLO
  - Robustness Testing on Dataset of Road Vehicles
- Development of a Real-Time Relative Path Following Solution using Monocular Camera which Leverages Open Source Libraries
  - Development of Platooning Specific Sub-tracker Algorithm
  - Real-Time Camera/INS Only Following Evaluation

1.4 Thesis Outline

Chapter 1 introduced the problem addressed in this thesis as well as adjacent work which deals with similar aspects of the problem. Chapter 2 covers the necessary background required to conceptualize the development of the algorithms presented in the thesis, as well as some information which could prove useful in continuing the work. Chapter 3 discusses the preliminary study which evaluates the open source object detection algorithm in various

road conditions, with varied cameras and ranging models. Chapter 4 covers the development of the real-time implementation of the methods discussed in Chapter 3, and the necessary development of the sub-tracker algorithm. Chapter 5 provides the conclusion of the work, and discusses ideas for improvement of the algorithms in order to guide future development.

Chapter 2

Background

2.1 CACC Platform

Over roughly the past decade, Auburn University's GPS and Vehicle Dynamics Laboratory (GAVLAB) has developed a system that serves as a testing platform for platooning of heavy duty trucks. The system was created largely by Grant Apperson [34], and the lineage continued with Patrick Smith [35] and Jake Ward [33]. The following section is a synopsis of the platform presented in their respective works, and serves as a framework for conceptualizing the work done for this thesis.

This cooperative adaptive cruise control (CACC) system consists of two Peterbilt 579 commercial trucks which can be seen below in Figure 2.1. The system implements longitudinal control and lateral control in order to allow for various platooning operations. Additionally, the system offers features such as cut-in detection and Connected Automated Vehicle (CAV) merging.

Figure 2.1: Peterbilt 579 Commercial Trucks.

The system is designed in such a way that either controller can be operated independently- i.e. testing can be performed using only the longitudinal controller or only the lateral controller. The distinction between the two is represented in Figure 2.2. When only using

one control mode, the system is classified as SAE Level 1, which implies that the function not performed by the system is done manually by the driver of the truck. However, when the system uses both control modes for operation, the system is classified as SAE Level 2, requiring the driver to only monitor for system error.

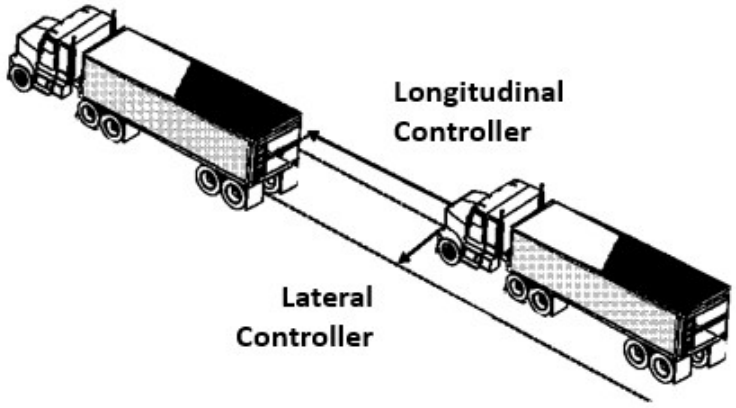

Figure 2.2: Lateral and Longitudinal Controllers.

2.1.1 System Components

Within the CACC system, there are three main components: the Dedicated Short Range Communication (DSRC) network, the Upper Level Control, and the By-Wire kit. Each will be discussed below in more detail.

DSRC

DSRC radios are used for Vehicle 2 Vehicle (V2V) communication between the trucks. The radios have an operating frequency of 5.9 GHz, and are considered to be the industry standard for this type of application. They serve as the method for passing state information between the trucks including vehicle speed, acceleration, and brake status. These states are then in turn used for the cooperative control algorithms. A custom User Datagram Protocol (UDP) packet is used in order to relay this information between each truck. Each UDP packet contains a vehicle ID, which allows for the source of the data to be identified. All vehicles

on the DSRC network have the capability to receive information from all other vehicles on the network. One major advantage to the implementation of the DSRC communication, is its flexibility, as it allows for the easy testing of new control architectures. For example, the UDP packet can be modified such that it contains new data, or edited data fields for use in such architectures.

Upper Level Control

The objective of the Upper Level Control system is to follow the lead vehicle at the specified distance via longitudinal control and along a relative path via lateral control. The Upper Level Control System is home to all of the algorithms and software libraries necessary for the CACC platooning operations. The data passed by the DSRC network is used for the estimation and control algorithms in this system. The output of the system are the commands sent to the vehicle for actuation: desired engine torque (longitudinal), engine retarder (longitudinal), break rate (longitudinal), and desired steer angle (lateral). Detailed descriptions of the control systems are provided later in this section.

By-Wire Kit

The By-Wire kit is the interface through which the software interacts with the vehicle, facilitating autonomous control. It operates by using a physical Controller Area Network (CAN) connection, which follows the SAE J1939 standard for heavy duty trucks. Through the software interface, data can be read and written to the CAN bus. Sensor data is read from various Engine Control Units (ECUs), which are built into the truck. This data includes current gear and brake status, among other readings. Not only can the By-Wire kit get readings from the various ECUs, it can send CAN messages to respective ECUs in order to implement the controller outputs. For example, the controller outputs a desired engine torque command and the By-Wire kit generates the appropriate CAN message. The CAN message is then sent over physical J1939 connection to the engine ECU. This is not unlike

the ACC which comes standard on the Peterbilt trucks, as the operating principles are fundamentally the same. In fact, the CACC system uses some parts of the ACC which were reverse engineered, including specific CAN message content and locations.

2.1.2 Hardware and Software Configuration

The CACC system consists of a number of hardware components and include an extensive sensor suite which works in tandem with an industrial grade computer which is designed to withstand rough conditions. The full hardware configuration can be seen below in Figure 2.3 and a list of each individual component and the role it plays in the overall system is also provided.

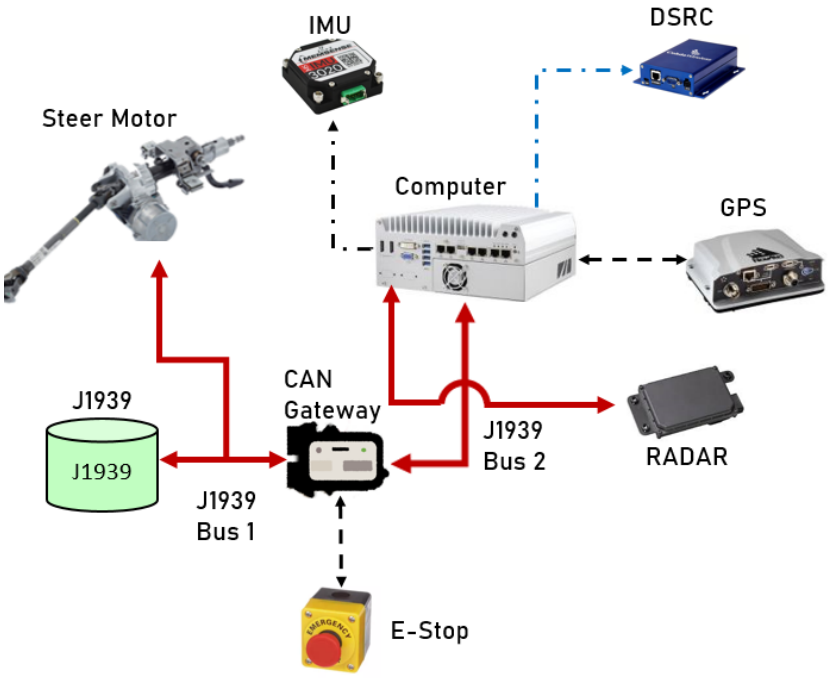

Figure 2.3: Hardware Configuration for CACC Platooning Platform.

Computer - Linux based unit which serves as the central component for all hardware operations. It receives all sensor and CAN bus data, and additionally the system software is compiled and executed on this computer.

DSRC Radios - Discussed in detail in Section 2.1.1, used for truck to truck communication.

GPS - Dual frequency GPS receiver, which is used for position information. Its output is also used for differential GPS relative positioning algorithms.

RADAR -Mounted on the front of the truck, the forward facing RADAR is used to provide high frequency range, range rate, and angular offset (bearing) measurements. It is also used for cut in detection of neighboring vehicles.

CAN Gateway - Connected between the computer's CAN bus and the rest of the vehicle, the CAN Gateway serves as a physical disconnect between the CACC system and vehicle.

Emergency Stop Button - Connected to the CAN gateway, if the button is pressed, manual control is returned to the driver.

Steering Motor - Commercial Steering motor which is integrated into the steering column of the Peterbilt. This allows for steering commands from the controller to be executed in order to accommodate for lateral error.

IMU - MEMS Grade IMU which is attached to the sprung vehicle mass, providing accelerations and rotation rates which are used in some algorithms.

The sensor suite and computer communicate through the Robotic Operating System (ROS). ROS is one of the most popular middleware options, and is known for its flexibility and ease of use. Within ROS, individual componentsm, known as nodes, are written for individual tasks. This modularizes any system written in ROS, which has many advantages. ROS has many supported programming languages; however, C++ and Python are typically used. ROS also has built in functionality including plotting, visualization, and data recording. The network of nodes utilized by the CACC system software is represented graphically in Figure 2.4. The ROS nodes include sensor drivers, the control and estimation algorithms, as well as the J1939 vehicle interface. The diagram consists of nodes belonging both to the lead truck and the follower truck. The lead truck has notably fewer nodes than the follower truck. This is because the lead truck is only required to collect relevant data and transmit it to the following truck. The following truck is required to collect the same data, while also

processing the data through its control and estimation algorithms. In the figure, the lateral control system's nodes are denoted in red, and the longitudinal control system's nodes in green. The algorithms programmed into these nodes are detailed in the next section.

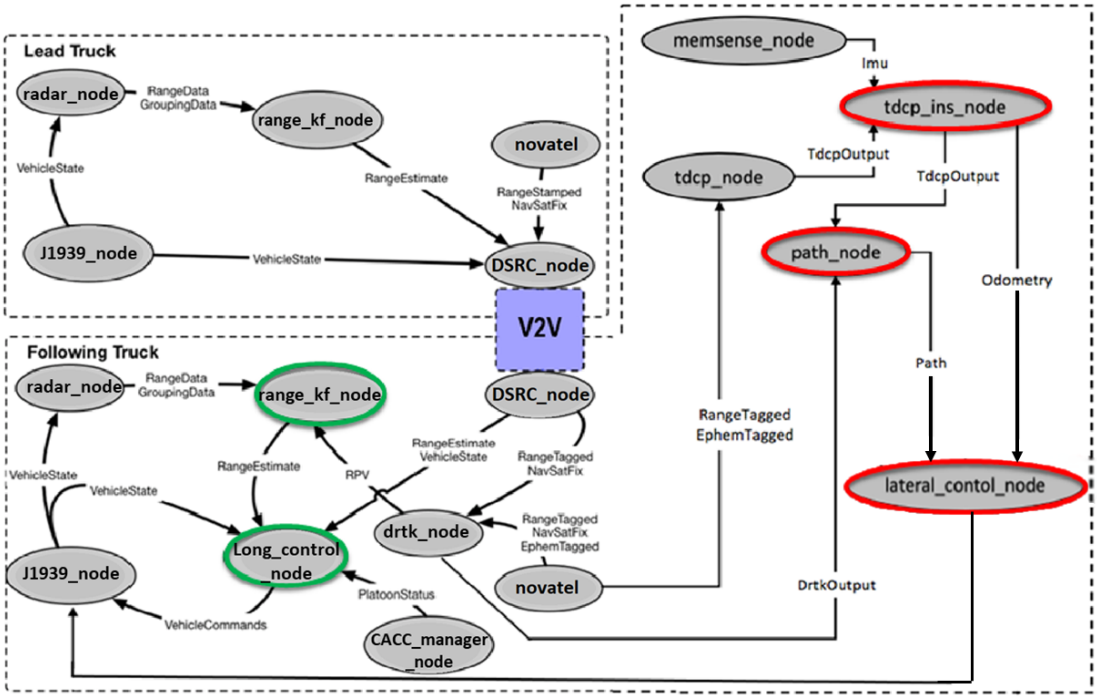

Figure 2.4: Hardware Configuration for CACC Platooning Platform.

2.1.3 Longitudinal Estimation and Control

The objective of the longitudinal controller is to produce commands to follow the lead truck at a specific distance. In order to accomplish this, estimation and control algorithms are implemented. The headway, or range, estimation algorithm combines multiple sensor readings in order to generate quality state estimates to feed to the longitudinal controller. The control system utilizes this estimate, along with a simple vehicle model and a classical controller, in order to track the reference gap distance. The schematic of the vehicle model is drawn in Figure 2.5. Using Newton's Second Law, from the free body diagram and driveline dynamics, the equation of motion can be determined to be the following:

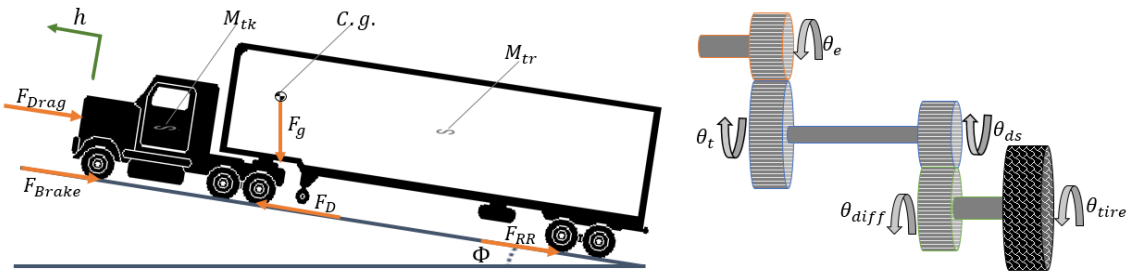

Figure 2.5: Diagram of Truck Forces and Gearing [34].

$$\mathrm{M}_{eff}\ddot{h} + b_{eff}\dot{h} = M_{eff}a_{lead} + b_{eff}v_{lead} - \frac{n_{diff}n_{trans}}{R_{eff}}T_{engine} + F_{brake} + F_{RR} + F_{air} + F_{grade} \quad (2.1)$$

The above equation makes the assumption that there is no wheel slip, as well as no driveline compliance. In Equation (2.1), h is the headway, and all of the forces expressed in the free body diagram can also be seen. The equation is specific to a following vehicle in the platoon and therefore, the dynamics of the lead vehicle can be incorporated into this model.

Range Estimation

The longitudinal controller relies on a high quality range estimate as the input. In order to produce this range estimate, a combination of relative GPS positioning and RADAR measurements is used. These sensor readings are used to produce states through the use of an estimation technique called the Kalman filter [34]. The Kalman filter is one of the most well-known methods in optimal estimation, and serves to combine measurements from various different sources.

DRTK

The relative position is determined by using a differential GPS technique called Dynamic Base Real Time Kinematic (DRTK) [5]. The DRTK algorithm uses the GPS carrier phase measurements, which are highly accurate, to determine a Relative Position Vector (RPV) between the vehicles. This technique is a step beyond RTK, which employs a fixed base-station in order to generate accurate global positons. DRTK still uses two receivers;

however, the "base-station" is now mobile, as it is located on one of the platooning vehicles. The GPS readings are communicated between the vehicles using the DSRC radios. One major advantage to DRTK is the fact that many common mode errors between GPS observables can be canceled out, thus improving the relative accuracy. The single differenced observables are then used in the measurement update of another Kalman filter to estimate the integer ambiguity of the single differenced carrier phase measurements. Following the resolution of the integer ambiguity, the carrier phase measurements are used to estimate the RPV between trucks using a weighted least-squares estimate. This RPV is an antenna-to-antenna measurement, as shown in Figure 2.6. After taking antenna offsets into account, the longitudinal and lateral component of the 3-D RPV can be determined. In addition, angular offset, or bearing angle between the vehicles, can be determined from these components.

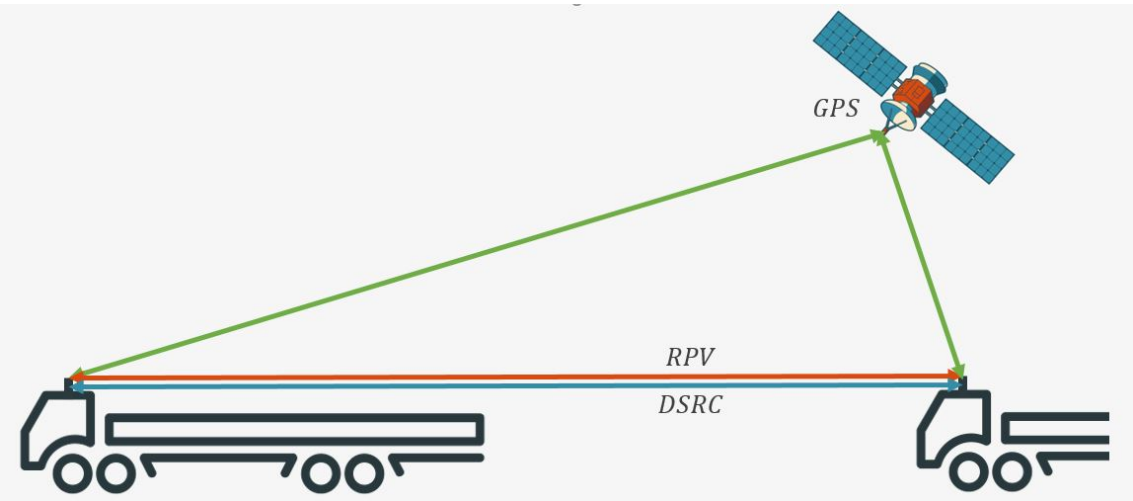

Figure 2.6: DRTK Algorithm Representation.

RADAR

A redundant measurement is gleaned from a forward facing RADAR that generates higher frequency range measurements. The Delphi ESR RADAR used by the CACC system has 64 channels, meaning for each epoch 64 data points are returned to the sensor. The RADAR has a range of 60m with a $\pm 45^{\circ}$ field of view. The output of the RADAR is range,

range rate (relative speed), and angular offset of a set of points perceived by the RADAR. An example of RADAR readings can be seen below in Figure 2.7, which also shows the perspective from the following vehicle's camera.

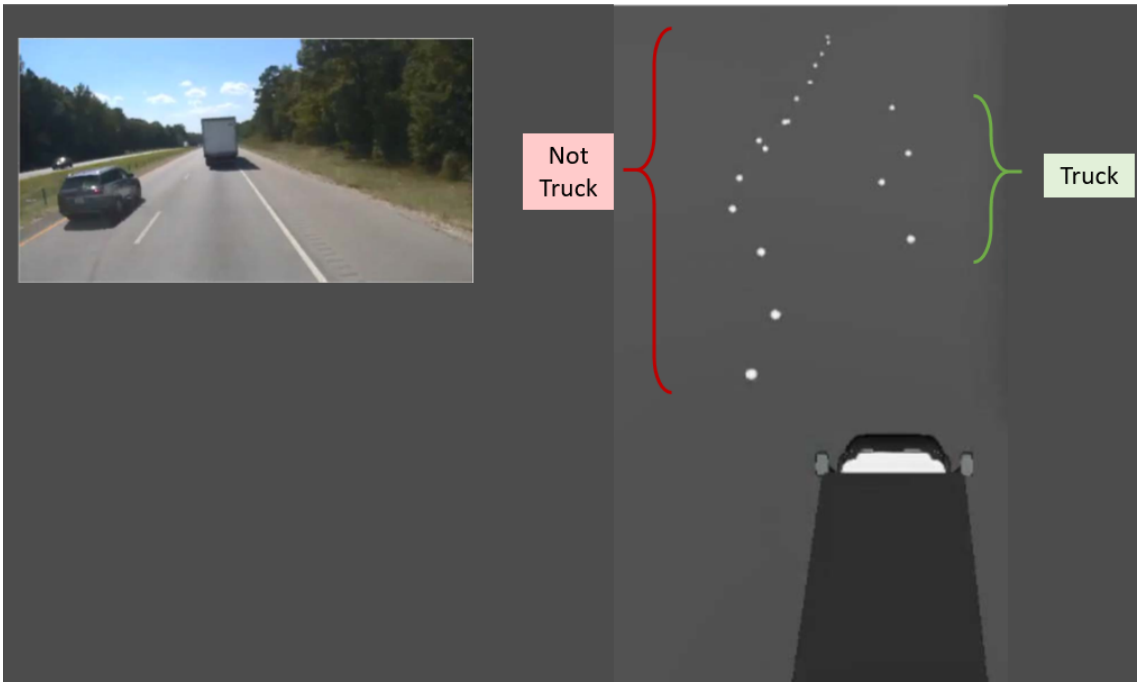

Figure 2.7: Visualization of RADAR Data and Accompanying Image [35].

From the figure, it can be seen that only certain data points are viable for determining the range to the lead truck. Some of the returned data points are associated with the car on the left as well as the guard rail which runs alongside the road. These additional measurements must be filtered out in order to have an accurate range determination to the lead truck. It should also be noted that points returned from the truck might be from various points on the truck, i.e. under the trailer, or the axles/tires. Valid points are considered to be those returned from the back of the trailer. In order to determine which points are valid, a Chi Squared test is implemented. The fundamental working principle of the Chi Squared test can be seen in Figure 2.8. The Chi Squared test considers the current state estimate and the measurement covariance, in order to determine which measurements are

valid and should be used as measurement updates in the estimator. Measurements outside of the measurement covariance ellipse are considered to not belong to the lead vehicle.

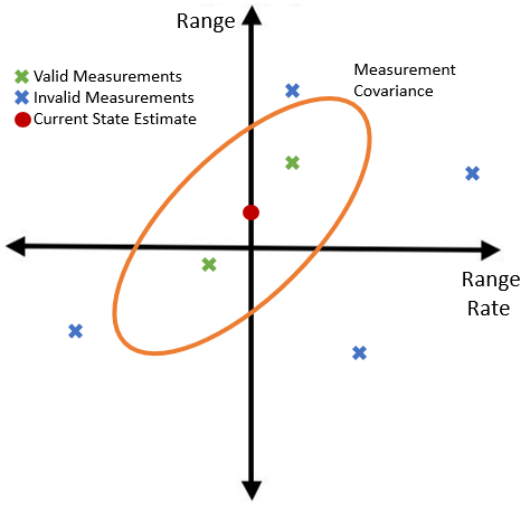

Figure 2.8: Representation of Chi Squared Test [34].

Kalman Filter

As previously mentioned, the Kalman filter is used to combine measurements from DRTK and RADAR. The filter outputs an estimate of the range/headway between platoon trucks, range rate, and angular offset. Figure 2.9 shows a schmeatic which represents the sensors utilized and desired states that are estimated. One assumption made by the filter is that range rate and bearing angle remain constant between measurement epochs. The time update of the filter is 20 Hz. In addition to predicting the state estimate, the filter also predicts the estimate covariance. When a new measurement is available, the measurement update is applied depending on the source. For DRTK measurements, the measurement update includes range and angular offset only. If the RADAR returns values that pass the Chi Squared test, the RADAR point produces a full state update (range, range rate, and angular offset). In the event that the points do not pass the test, points belonging to the truck can still be used to determine the range rate, or relative speed between the two vehicles.

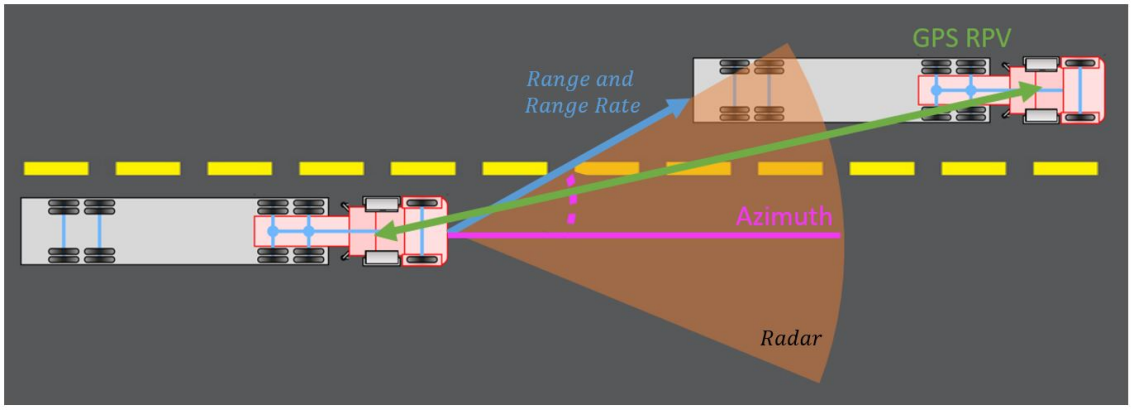

Figure 2.9: Kalman Filter Input Measurements and Output States (Range, Range Rate, and Azimuth) [35]].

Longtiudinal Control

The longitudinal controller serves to control the distance between the platoon trucks. The controller uses a classical PID (Proportional Integral Derivative) controller with feed forward components. Figure 2.10 shows the schematic of the longitudinal controller. The controller computes headway error (red h) from estimated range (blue h) and desired headway (h_{ref}). A feed forward torque T_{FF} is also considered by the controller and is used to correct for known determininstic disturbances in the longitudinal model [35].

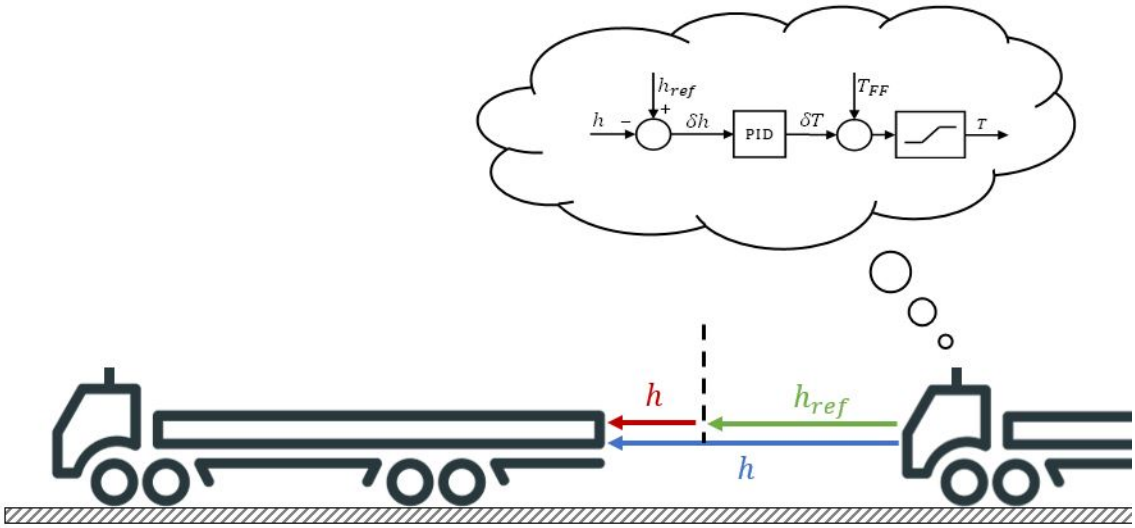

Figure 2.10: Longitudinal Control Scheme [35].

2.1.4 Lateral Estimation and Control

System Description

The goal of the lateral control structure is to closely follow the path of the lead truck. The lateral following method has two components. First a way point generation algorithm which generates GPS points along the path taken by the lead vehicle. The second component is the lateral controller which exists to replicate this path as accurately as possible. This method is commonly called the "bread-crumb following" method, after the classic fairy tale "*Hansel and Gretel*" [81]. The lateral control structure uses the waypoints, vehicle model, and a classical controller that considers lateral error relative to the path.

TDCP

Time-Differenced Carrier Phase (TDCP) is a differential GPS technique that uses carrier phase measurements for a single vehicle, and is a method that is not necessarily platooning specific. The measurements are differences from time t_{k-1} to time t_k, to produce a high precision odometry measurement from a single antenna on the vehicle as shown in Figure 2.11. By assuming a short time step, the difference measurements are highly correlated, allowing the removal of many common sources of error, including integer ambiguity. The change in position of the receiver is estimated using weighted least squares. TDCP provides a robust odometery measurement, assuming that there is no cycle slip while receiving the carrier phase measurements. A mask angle is used to filter satellites closer to the horizon, in an effort to limit cycle slip.

Path Generation

The path generation algorithm combines DRTK and TDCP measurements to estimate a RPV to a virtual lead vehicle. The virtual lead vehicle position is a previous position of the lead vehicle from a previous GPS epoch, represented as orange dots in Figure 2.12. These

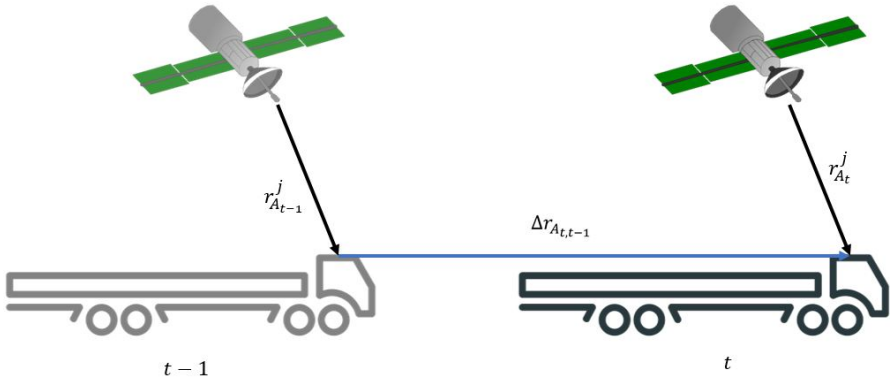

Figure 2.11: TDCP Representation [35].

waypoints are then followed by a classical PD (Proportional Derivative) controller which considers the heading error between the current heading and the desired waypoint. A block diagram of the controller can be seen in Figure 2.13.

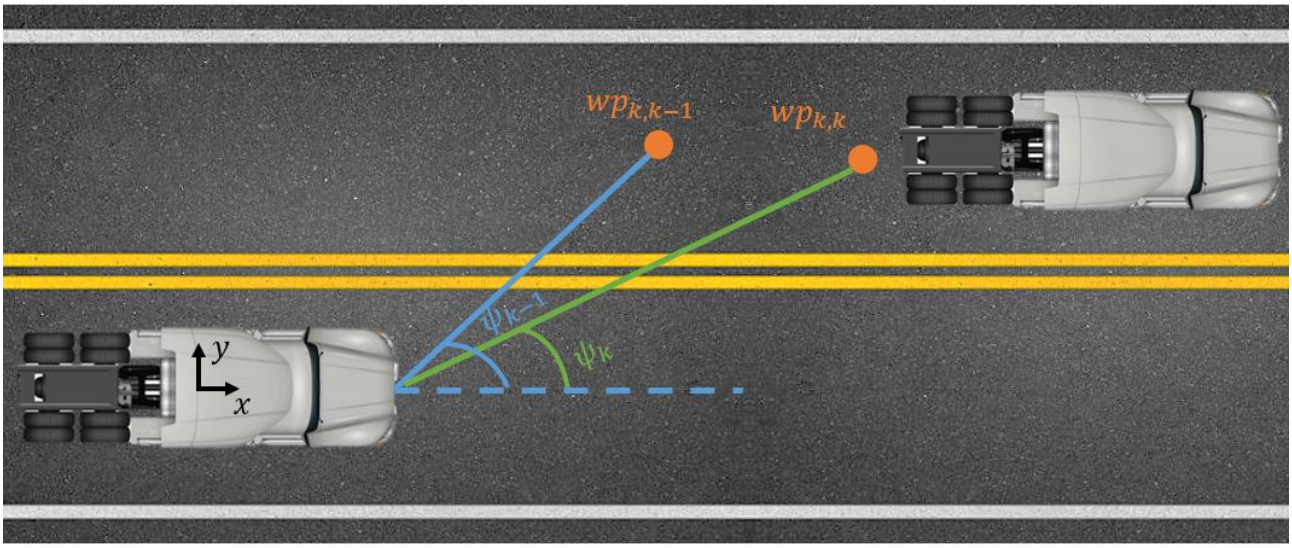

Figure 2.12: Graphical Representation of Waypoint Following Method.

The mathematical expression for the controller output for the desired yaw rate is seen in Equation (2.2):

$$\delta = k_p(\psi_{ref} - \psi) + k_d(\dot{\psi}_{des} - \dot{\psi}) \tag{2.2}$$

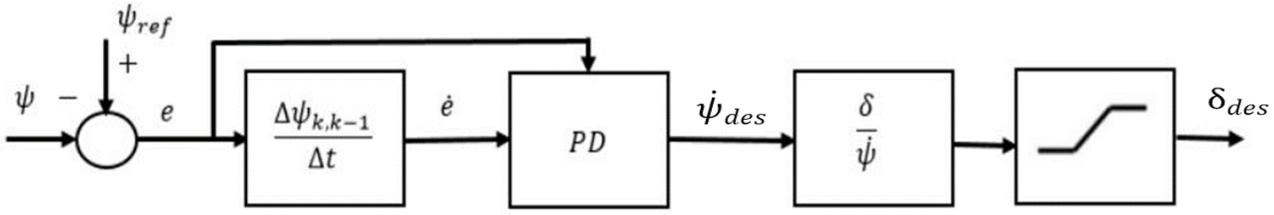

Figure 2.13: Block Diagram of Lateral Controller.

where $(\psi_{ref} - \psi)$ and $(\dot{\psi}_{ref} - \dot{\psi})$ are the error and derivative of error, respectively. The gain values k_p and k_d are determined experimentally through hand tuning, and a list of tabulated gains exists to handle a variety of trailer loads. The steer angle δ is the command sent to the steering motor for lateral actuation. Note that the desired steer angle is the steer angle at the tire. Therefore, the tire steer angle is multiplied by the known steer column gear ratio, converted to degrees, and passed through a saturation block before being sent to the By-wire kit for automated steering control.

2.2 Camera Background

2.2.1 Camera History and Working Principles

Historical Perspective

The first camera in recorded history is more akin to a modern projector than a modern camera. The camera obscura consisted of lenses and mirrors to create projections of scenery onto a whitened wall, as seen in Figure 2.14 which is an illustration of how the device was used to view a solar eclipse at Louvain on January 24, 1544. The technology proved useful as a scientific instrument, as well as an aid in tracing entities for artistic endeavours [32].

It was not until the 18^{th} century that Thomas Wedgewood was credited with being the first to reliably document the use of light sensitive chemicals to record silhouette images on durable media (i.e create the first known photographs) [36]. He is also the first known individual to attempt photographs using projections created by a camera obscura. It is

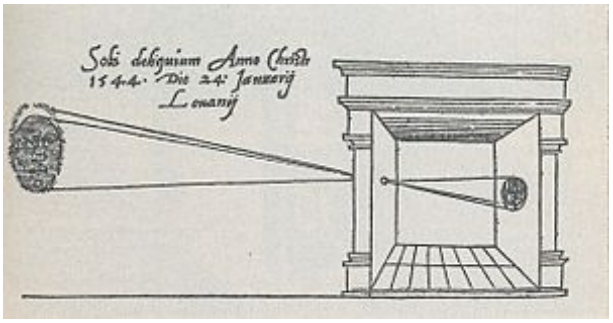

Figure 2.14: Image of the Camera Obscura from *De Radio Astronomica et Geometrica* circa 1545 [65].

believed that there are no remaining photographs made by Wedgewood, due to the fact that they remained light sensitive following printing, and would become ruined. Work continued by other chemists made the photographs light stable after the initial process, and also reduced the exposure time, making the methods easier to implement. The result was the earliest known surviving photograph, 'point de vue', which is displayed in Figure 2.15. The first commercially available camera was known as the Daguerreotype [37]. It formed images onto silvered copper plates that were developed with mercury vapor. The cameras could take up to half an hour to expose light to the plates.

Figure 2.15: Joseph Nicéphore Niépce (French, 1765–1833), Untitled 'point de vue,' 1827. - The earliest known surviving photograph taken with a camera obscura [66].

The use of film based photography was pioneered by George Eastman, who released his first camera, the "Kodak", in 1888 [38]. The first capture of motion using cameras also occurred around this time, resulting in the movie industry by the end of the 19^{th} century. Film proved to be an effective medium for photography, with improvements in quality, portability, and affordability. As of July 2023, it is still possible to purchase a disposable film camera at the Auburn Walmart, proving the lasting power that film photography has had.

Digital cameras function in a similar way to traditional film cameras, except instead of light being focused onto a light sensitive chemical, the light is focused onto a photosensitive electronic device. The earliest semiconductor image sensor was the charged-coupled device (CCD) invented at Bell Laboratories in 1969 when researchers realized that an electric charge could be stored on a small Metal-Oxide Smeiconducter (MOS) capacitor [39]. Figure 2.16 shows an oscilloscope output of the first CCD image, and it can be seen that the resolution was very low. The MOS capacitors could easily be arranged into arrays. Advances in MOSFET technology allowed for the production of the Complimentary Metal-Oxide Semiconductor (CMOS) in the early 1990's. It is believed by many that the first true digital camera commercially released was the DS-X by Fuji in 1989 [40]. There is confusion surrounding this due to the limited number produced and the associated high cost.

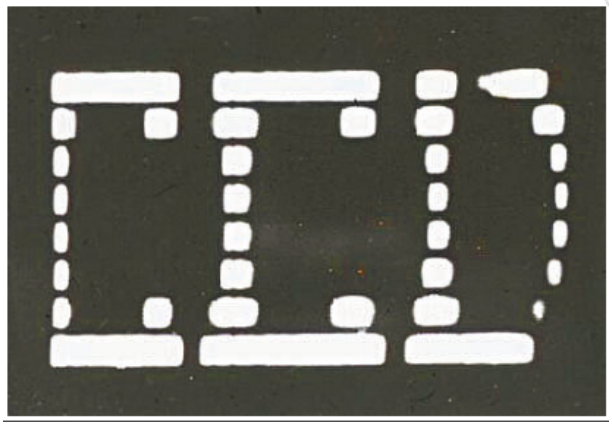

Figure 2.16: Oscilliscope Output from the first CCD Imager [67].

Modern Camera

As mentioned in the previous section, modern cameras function in a similar way to traditional film cameras. Figure 2.17 shows the basic components of a modern camera. The lens represented on the left is functionally the same as a film camera and serves to focus light onto a photosensitive electrical device known as *Imagers*. The imager converts the light measurement into an electrical signal which is processed on a PCB.

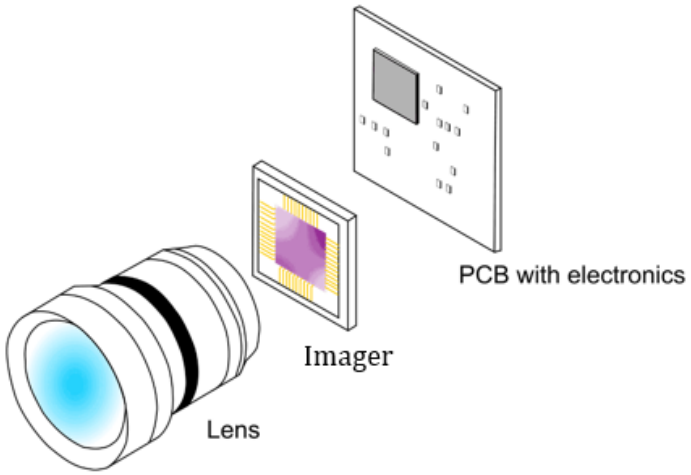

Figure 2.17: Structure of the Modern Camera [68].

There are two main options when it comes to to imagers: Charged Coupled Device (CCD) and Complementary Metal-Oxide Semiconductors (CMOS) [41]. Functionally, the two imagers are the same. Each can be thought of as a grid of elements, with each grid element being capable of converting light into electrons by capturing light photons. Each grid element is refereed to as a photosite. When an image is being recorded, the grid is exposed, allowing the charges to accumulate. Then, electrical charge at each element is quantified, which is where the two imagers differ. The difference in the working principles between CCD and CMOS imagers is shown in 2.18. For the CCD, the charge is read at one corner of the grid, after being transported across the chip. An analog-to-digital converter turns each photosite's charge into a digital value. In CMOS based devices, each photosite

contains multiple transistors which serve to amplify and move the charge independently. This allows flexibility with regards to applications, because each charge can be read individually.

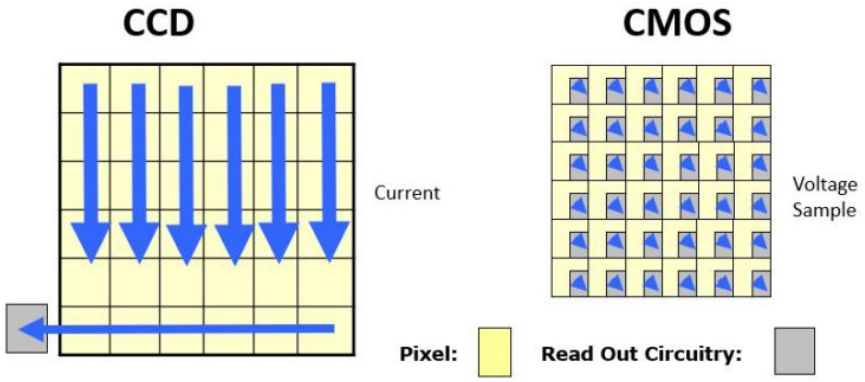

Figure 2.18: CCD v CMOS Working Principle [69].

Specially manufactured CCD devices allow for the ability to transport charges across the chip without distortion, creating high-quality, highly sensitive sensors. CMOS on the other hand, are easier to manufacture, capable of being manufactured on most standard silicon production lines. While CCD sensors create comparatively high-quality, low-noise images, they are traditionally more expensive. Additionally, CCD sensors can consume as much as 100 times more power than a comparable CMOS sensor. The current state and potential future of the decision making process regarding CCD vs CMOS could be summarized as follows. The best quality images come from a CCD camera; however, this performance comes at a cost, both monetarily and required power. Older CCD cameras tend to be frequently replaced by CMOS cameras. The performance gap between CMOS and CCD cameras is being closed, as advances in CMOS technology continually improves performance. It is expected that eventually CCD cameras will be phased out by CMOS.

Photosites' Perception of Color

An important aspect of understanding the working principles of modern cameras is how color is perceived by the camera. Each photosite in the imager array is incapable of perceiving color, and can simply capture the intensity of the light hitting it. To compensate,

light is filtered based on the primary light colors (Red, Blue, and Green). This is done using a color filter array. The most commonly used filter is the Bayer Filter Array which alternates a series of red, blue, and green in a checkerboard pattern [42]. A representation of a Bayer Filter is shown in Figure 2.19, and it can be noticed that there are twice as many green squares as there are red or blue. This is because in order to recreate images in a similar way to how the human eye perceives, it is necessary to replicate the fact that the eye is not equally sensitive to the three colors. Interpolation is used between the squares to determine the appropriate pixel values for 3 channel images. It should be noted that this is not necessary for single channel images which perceive in gray-scale.

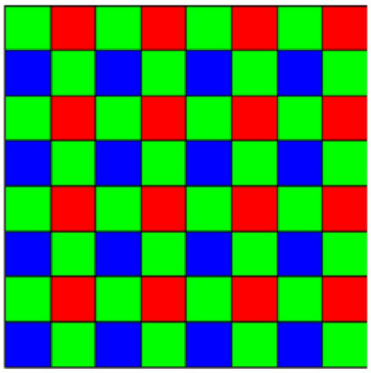

Figure 2.19: Bayer Filter Configuration [42].

Global vs Rolling Shutter

Another thing to consider when deciding on a camera for a specific application is rolling shutter vs global shutter [43]. The methodology behind rolling shutter is reminiscent of older cathode ray tube (CRT) televisions in which a laser passed over all locations on the screen creating scan lines due to the rolling shutter processing pixels one at a time. The method can create a stretching effect in dynamic situations. This is why for most computer vision applications, it is preferable to use global shutter which captures an image in its entirety without these effects. The effect of motion blur induced by the rolling shutter is shown in

Figure 2.20, in which the two different shutters are used to capture a dynamic system (a rotating pedestal fan).

Figure 2.20: Global Shutter and Rolling Shutter [70].

2.2.2 Pinhole Camera Model

Most introductory level computer vision literature begins by posing the question of accurately representing 3D objects in a 2D image, or more precisely, how is a camera modeled? A good resource for familiarizing with the subject is the open source class from Stanford which includes the following camera model derivations [44]. The simplest and easiest to conceptualize camera model is the pinhole camera model, which is illustrated in Figure 2.21. In this model, light is considered to be reflected onto the pinhole frame from the scene. However, only a singular ray of light from each point in the scene enters the pinhole. The purpose of the barrier is to prevent every point on the 3D object from influencing every point on the object. The barrier causes the creation of a one-to-one mapping between the 3D object and the 2D image plane.

A more complete representation of the pinhole camera model is shown in Figure 2.22. Here, the aperture point is considered to be the pinhole O (this is sometimes referred to as the center of the camera). The distance between the image plane and the pinhole is considered to be the focal length f. It can be seen that point P on the 3D object is projected to point P' in the image plane. Here $P = [\ x \quad y \quad z\ ]^T$ and $P' = [\ x' \quad y'\ ]^T$. The projection of the pinhole O onto the image plane is represented by C'. A coordinate system $[i \quad j \quad k]$ is

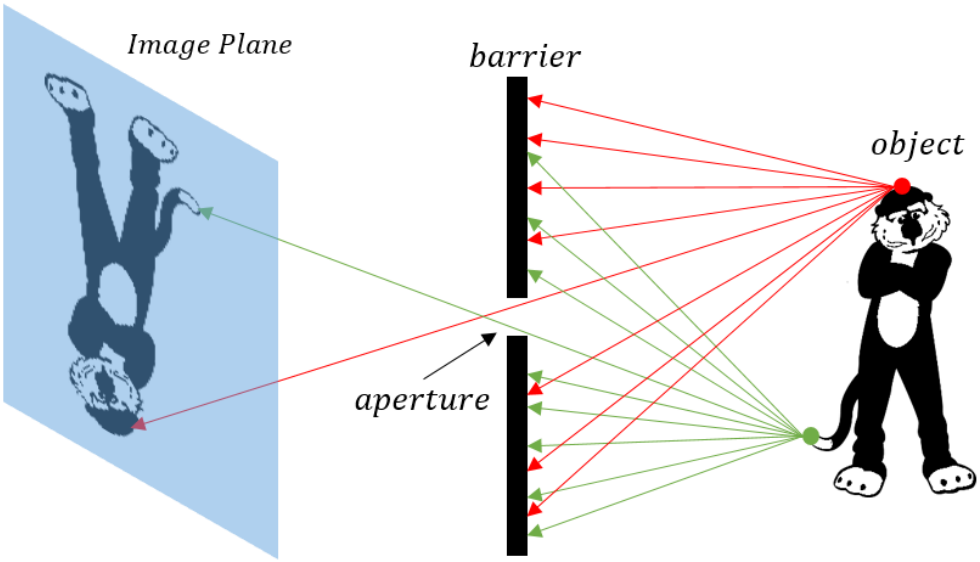

Figure 2.21: The Pinhole Camera Model.

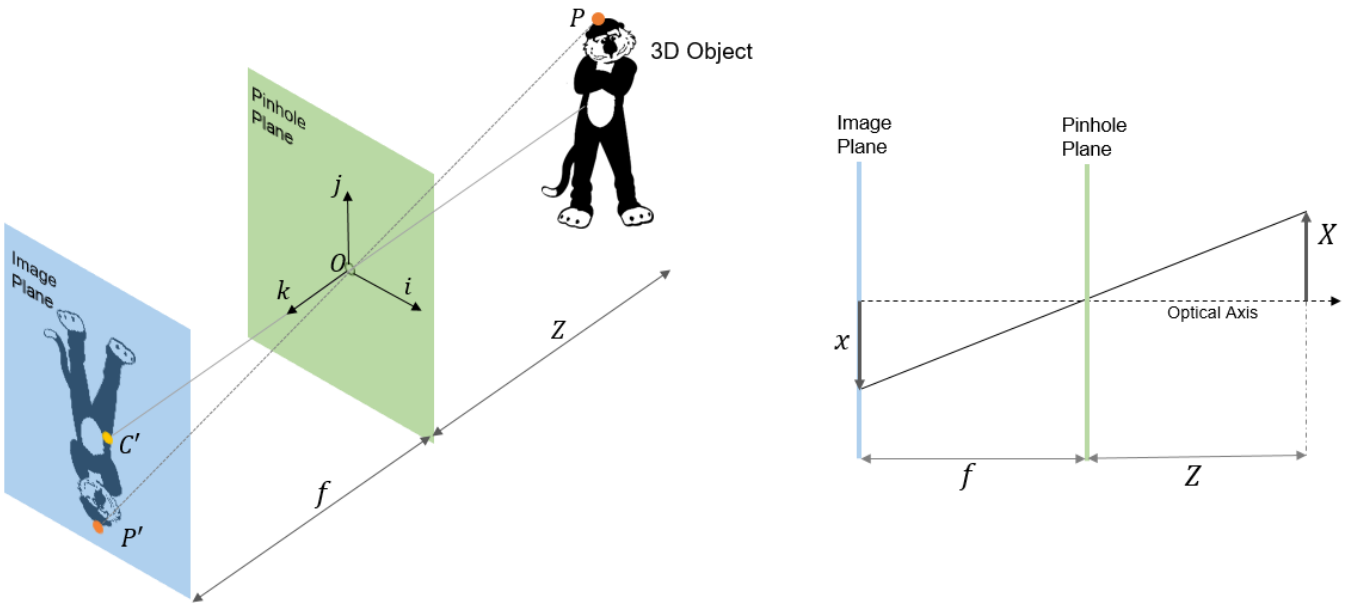

Figure 2.22: Formal Representation of the Pinhole Camera Model.

defined and has its origin at the pinhole. This coordinate system is commonly referred to as the camera coordinate system. The line between O and C' is collinear with the k axis of the camera coordinate system. This line is referred to as the optical axis. It can be seen in Figure 2.22 that triangle $P'C'O$ and the triangle defined by P, O and (0,0,z) are similar triangles. Utilizing the law of similar triangles, it can be seen that:

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{bmatrix} \tag{2.3}$$

An assumption made by the pinhole camera model is that the aperture is a single point; however, this assumption fails in that the aperture cannot be infinitely small. The effects of increasing the aperture size can be seen in Figure 2.23. It can be seen that as the size of the aperture increases, the sharpness of the image decreases. Additionally, the image becomes brighter. Both of these effects are the consequence of more light rays passing through the aperture. This is a fundamental problem with pinhole cameras in that there is a trade-off between image crispness and brightness.

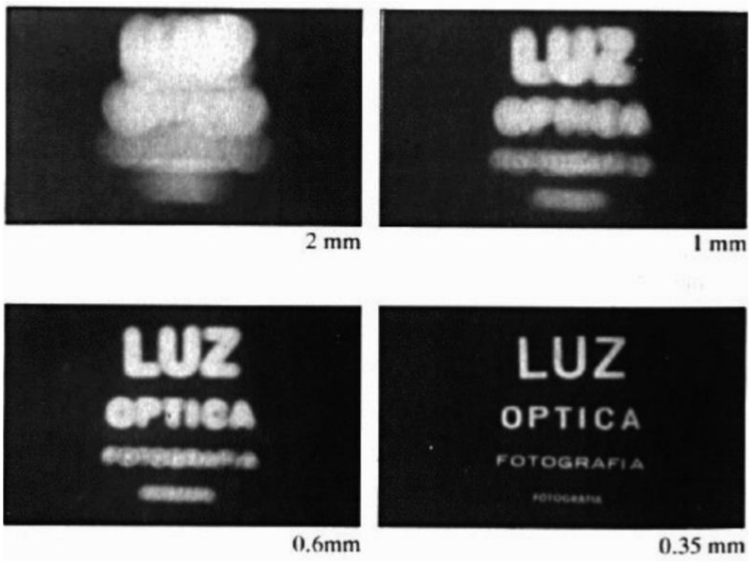

Figure 2.23: The effect of aperture size on the resulting image [18].

2.2.3 Cameras with Lenses

The trade-off between image crispness and brightness is mitigated in modern cameras through the use of lenses, which, when used appropriately, can focus all light rays emitted by a singular point in a scene to a single point on the image plane. This can be seen graphically represented in Figure 2.24. The property does not apply to all 3D points. For example, points that are too close or too far away will be blurred or out of focus. This concept is referred to as depth of field, which is the range at which cameras can take clear photos. This be seen in the figure, in that the light rays on Aubie's tail are converging; however, the light rays on Aubie's tail do not converge because of the depth of field

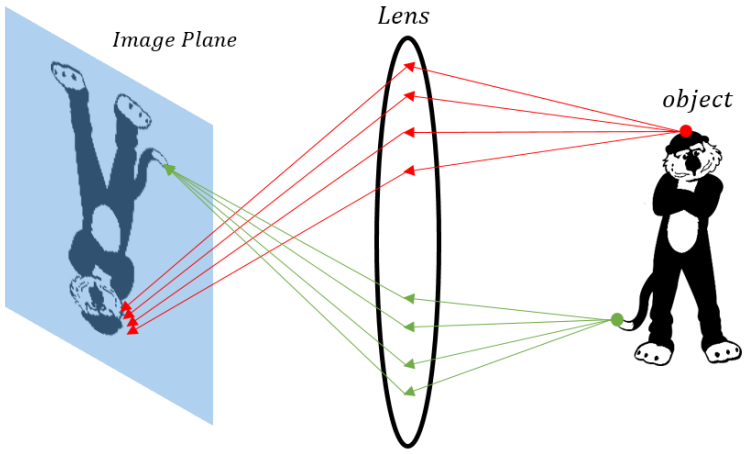

Figure 2.24: A representation of a lens model.

Lenses also focus all light rays traveling parallel to the optical axis onto a singular point called the focal point. A modification to the pinhole camera model can be made which considers lens usage can be seen in Figure 2.25. The distance between the focal point and the centroid of the lens is referred to as the focal length, f, of the lens. The modification of the pinhole camera model can be adapted to include the lens model by considering the distance between the object and the focal point, and the focal length.

Utilizing the "thin lens" or paraxial assumption, it is possible to derive a model that relates point P in 3D space to a corresponding point P' in the image plane. For the pinhole

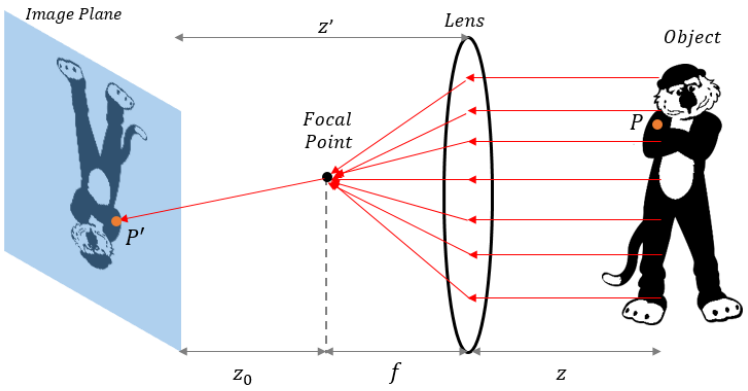

Figure 2.25: Light rays that enter the lens parallel to the optical axis converge on the focal point.

model $z' = f$; however, for the paraxial refraction model $z' = f + z_0$. The paraxial refraction model can be seen in Equation (2.4):

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} z' \frac{x}{z} \\ z' \frac{y}{z} \end{bmatrix} \tag{2.4}$$

It should be noted that within the derivation for the paraxial refraction model, the lens is approximated as thin. This causes issues when used on real world lenses. The most common issue is radial distortion, which causes magnification to increase or decrease as a function of the radial distance from the center of the lens. Examples of radial distortion can be seen on the image of a checkerboard in Figure 2.26. This is seen as either pincushion distortion (radially increasing magnification) or barrel distortion (radially decreasing magnification). These effects are caused by different regions of the lens having different focal lengths.

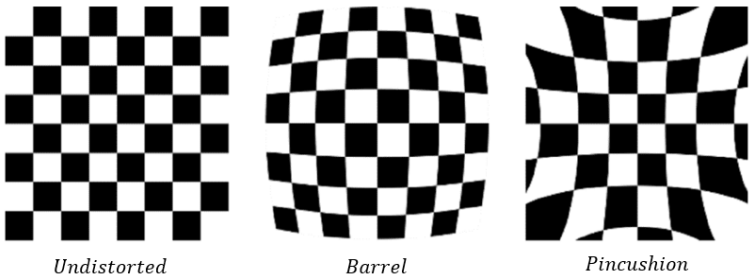

Figure 2.26: Depiction of barrel and pincushion radial distortions [71].

2.2.4 Digital Image Space

The mapping of a point P in 3D space to a point P' in 2D space is referred to as a projective transformation. There is not a direct correspondence of the projection of 3D points into the 2D image plane represented in digital images. There are a few reasons for this. The first is that digital images are discretized, while the image plane itself is continuous. Additionally, points in digital images are generally in a different reference system than the image plane. Also, there are many undesirable effects caused by the non-linear nature of physical cameras. Through the use of transformations it is possible to mitigate these issues, and directly map points in a 3D environment into the digital pixel coordinates.

Camera Matrix Model

When discussing the pinhole and paraxial refraction model, the coordinate system used for these derivations has an origin along the optical axis (image center). Computer vision algorithms traditionally have their origin in the upper left-hand corner of the image. To account for this, two parameters are introduced to the pinhole camera model, c_x and c_y. These parameters account for the translation between the image center and the desired digital image coordinate system. Another effect that must be considered is that in the image plane, points are expressed in units of physical length, whereas in digital images, these points are expressed in pixel coordinates. To account for this transformation, another two parameters are introduced k and l. These parameters have units of $\frac{pixels}{length}$. It should be noted that k and l are not necessarily the same due the fact that pixels are considered to be square and it is common to see images that are rectangular (1280x720px, 640x480px, etc.). The modified pinhole camera model, which has these new terms can be seen in Equation (2.5):

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} fk\frac{x}{z} + c_x \\ fl\frac{y}{z} + c_y \end{bmatrix} = \begin{bmatrix} \alpha\frac{x}{z} + c_x \\ \beta\frac{y}{z} + c_y \end{bmatrix} \tag{2.5}$$

By utilizing a homogeneous coordinate system, it is possible to derive the camera intrinsic matrix K which organizes the camera parameters into a single 3x3 matrix. It is also sometimes referred to as the camera matrix. It allows for the projection of the points in a 3D environment into the camera. This can be represented by the simple form shown in Equation (2.6), and is valid for when skewness (angular offset between the imager and lens) can be considered negligible:

$$P' = \begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.6}$$

Extrinsic Parameters

The formulae up until this point have been considering the 3D camera reference system. However, it is possible to use another transform which relates points from the world reference system to the camera reference system. This is done by utilizing a transformation which considers the rotation matrix R and translation vector T. If a point in a world reference system is given by P_w, it can be determined in camera coordinates by Equation (2.7):

$$P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w \tag{2.7}$$

Combining this with our intrinsic parameter model provides:

$$P' = K \begin{bmatrix} R & T \end{bmatrix} P_w \tag{2.8}$$

2.2.5 Camera Calibration

In order to properly use the transformation from 3D environments to 2D images, it is necessary to have accurate knowledge of the camera's intrinsic parameters. These parameters are either difficult or impossible to measure. Additionally, even two identical cameras from

the same manufacturer might have different intrinsic parameters. Parameters could change over time, especially if a camera is dropped or exists in a rugged environment. While it may not be possible to access these parameters, it is possible to estimate them using images from the camera. This process is known as camera calibration and involves taking numerous images of a known pattern (typically a checkerboard) at different orientations. There are open source libraries that can handle this process for the user, including OpenCV which has functions specifically for this application. The ROS camera calibration tool can be seen in Figure 2.27. The ROS implementation which features a GUI displays information about the captured poses, and provides feedback for which poses have not yet been captured [45]. Additionally, after calibration, the GUI displays the calibrated and uncalibrated versions of the image data. This allows the user to ensure a visually accurate calibration. The tool also allows the user to save the camera calibration parameters to a YAML file which can then be used by the camera specific ROS driver in runtime.

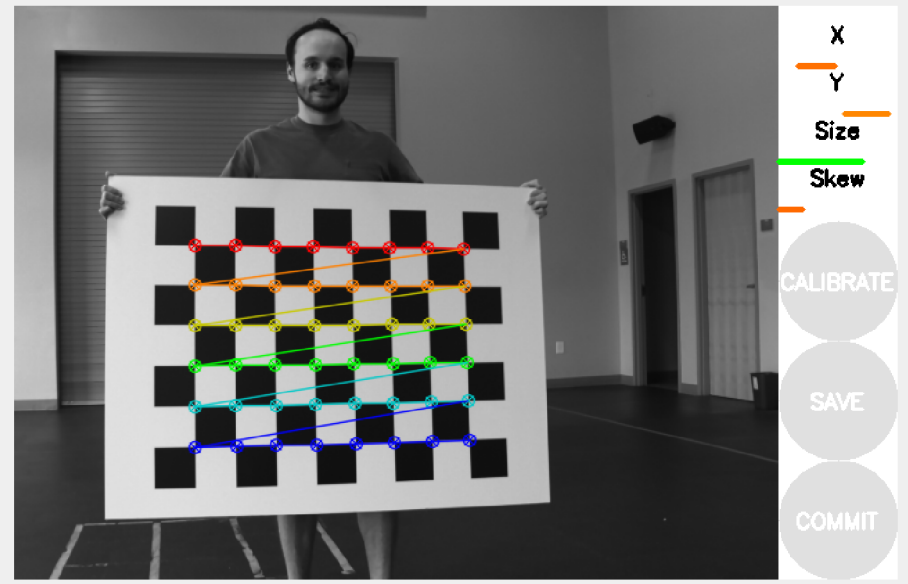

Figure 2.27: ROS Camera Calibration Tool [45].

There are many derivations of the camera calibration method available online [46]. The central idea is that lens distortion (radial and tangential), can be modeled as a polynomial and can then be used to remap pixels to their correct location. The polynomials are a

function of the radius, r, where $r = \sqrt{x^2 + y^2}$, and x and y are the coordinates for each pixel in an image. The polynomials for radial distortion are seen in Equations (2.9-2.10):

$$x_{corrected} = x \cdot (1 + k_1 r^2 k_2 r^4 + k_3 r^6) \tag{2.9}$$

$$y_{corrected} = y \cdot (1 + k_1 r^2 k_2 r^4 + k_3 r^6) \tag{2.10}$$

Tangential distortion can be caused by the lens not being fully parallel to the imager (image plane). This is represented in Figure 2.28 where the top of the lens is further away from the imager than the bottom of the lens. This causes the distortion in the image on the right.

Figure 2.28: The Cause and Effect of Tangential Distortion

Tangential distortion can be modeled in a similar way to radial distortion, as given by Equations (2.11- 2.12):

$$x_{corrected} = x + 2p_1 xy + p_2(r^2 + 2x^2) \tag{2.11}$$

$$y_{corrected} = y + p_1(r^2 + 2y^2) + 2p_2 xy \tag{2.12}$$

Equations (2.9-2.12) contain a sum of 5 unknown variables which can be solved using the set of detected vertices from the checkerboard and a least squares method. The results of calibration can be seen on two images in Figure 2.29. The first pair of images shows the effect of lens distortion being removed following camera calibration. The vehicle behind the

barricade is seen to be more proportional, and the trees become more vertical. The second pair of images were included specifically to show the effect of lens distortion on rectilinear entities, and the removal by camera calibration

Figure 2.29: Difference between Calibrated and Uncalibrated Images.

2.3 Defining Computer Vision

Computer Vision is a widely growing field, growing at a rate never seen before. This is due to the continual improvement of processing speed, allowing for more algorithms to run on a wider variety of devices. Additionally, with the wide acceptance of smart devices, cameras are more accessible than ever. Many misconceptions about computer vision have come from the fact that humans possess the ability of sight, and assume that computers must process image data in a similar way to that of a human. Figure 2.30 represents how a human might perceive a scene in comparison to how a computer would perceive a scene. To a computer, image data is nothing more than a matrix or tensor. A working definition

of computer vision can be defined in this thesis as follows: *using pixel values to determine practical information of a scene a camera perceives.*

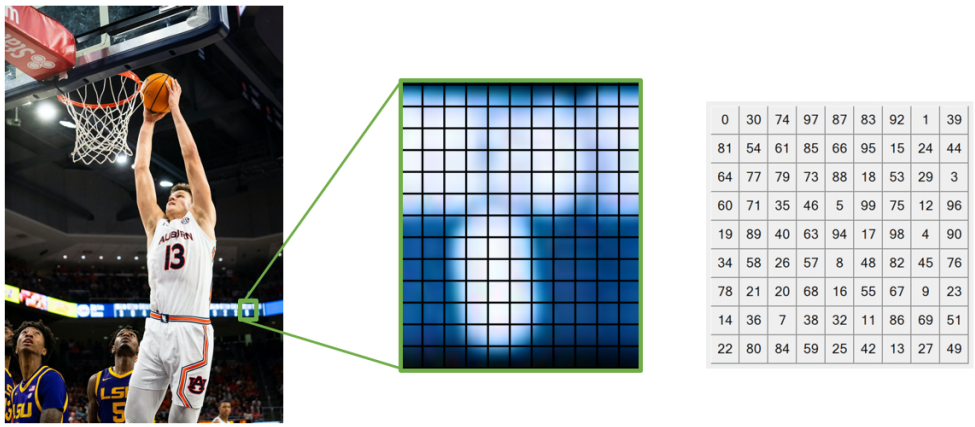

Figure 2.30: Representation of how computers see images.

In its current state, computer vision is worse than human visual perception in most ways. There are a few exceptions to this due to the ways in which the human brain utilizes contextual information in an image. This makes the human brain fallible to things such as the Checker Shadow Illusion seen in Figure 2.31 [47]. To the human brain, squares A and B appear to be different colors, when in fact they are the same color. However, a computer vision algorithm can easily tell that the squares have identical pixel values.

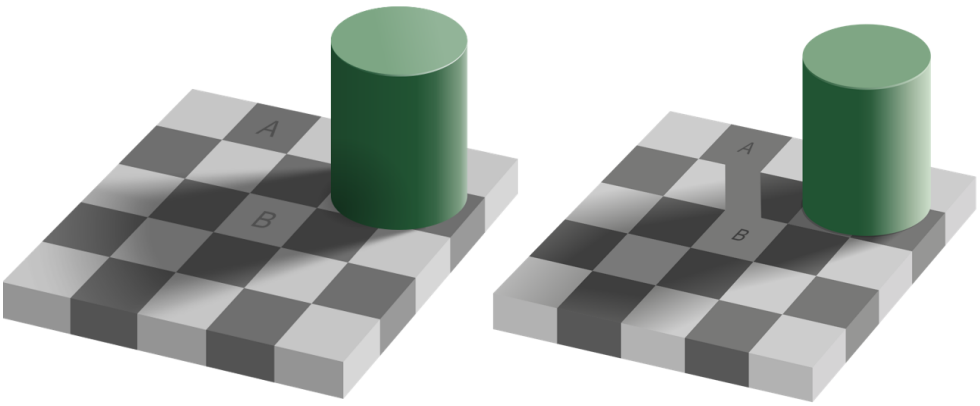

Figure 2.31: The famous checker shadow illusion [47].

2.3.1 Color Spaces

Following the use of the Bayer filter and interpolation, each pixel value is assigned a pixel value that corresponds to RGB values. This is the most commonly used color space, but there are other options which provide some benefits depending on the application [48]. Utilizing a different color space can be thought of as viewing an image in a different format in order to gain an advantage for a later algorithm, making something more detectable or easily discernible. There are many color spaces to choose from, and a few of the most popular color spaces will be discussed. Three representations of the most commonly used color spaces can be seen in Figure 2.32. One such color space is Hue Saturation and Lightness (HSL). HSL describes color as a bi-conical solid in 3 dimensions as shown in the right of Figure 2.32. The vertical axis describes lightness and is represented between 0 and 1. All fully saturated colors lie on the outer circumference where the two cones meet at a value of $L = 0.5$ and are described as an angular value to represent hue. The third value is saturation and is expressed as a radius around the vertical axis. Hue Saturation and Value (HSV) is another color space that is used in computer vision, and is represented as a 3D cone shape as shown in the right image in Figure 2.32. Hue and Saturation work identically between HSV and HSL; however, the Lightness term is exchanged for a Value term (sometimes referred to as Brightness) evaluated between 0 and 1.

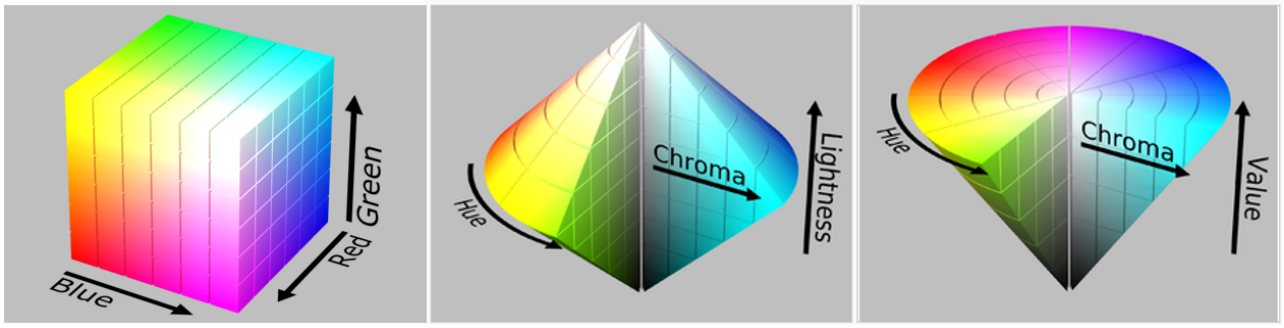

Figure 2.32: The most commonly used color spaces (left: *RGB*, center: *HSL*, right: *HSV*) [72].

There are a few benefits to using a colorspace other than RGB. The primary is that the differences between certain colors in RGB can become larger in another color space, making objects an scenery easier to discern. This can be seen in Figure 2.33 where certain elements become more evident in the image, especially shadows. Converting colorspaces has an application in object detection and tracking, and is used to detect objects or scenery based on color. It is commonly used in lane detection algorithms in order to filter for yellow or white lane markings. This will be covered in Section 2.3.5 detailing commonly used algorithms for autonomous vehicle applications.

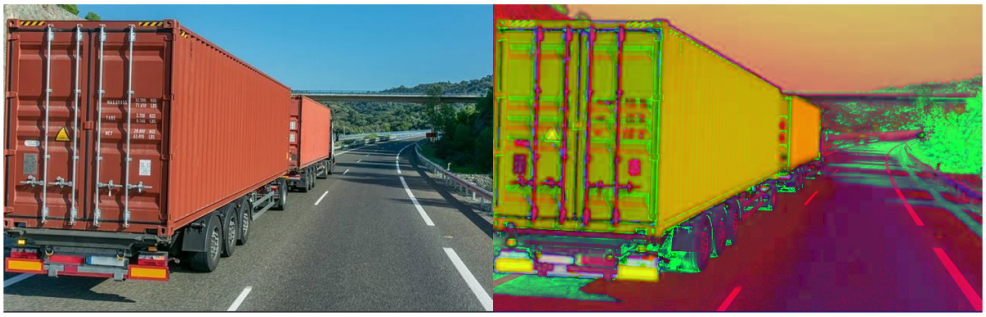

Figure 2.33: Comparison of RGB vs HSV color space.

Another popular color space is the LAB color space. It expresses colors in 3 channels: L for lightness, A for colors ranging from green to magenta, and B for colors ranging from blue to yellow. The color space is designed to closely emulate the way in which the human brain perceives color. Because the two color channels do not consider intensity, the color space is an excellent choice when filtering off of color alone, making algorithms more robust to variations in lighting. This can be seen in Figure 2.34, in which two photos of a Rubik's cube are split into the three channels of the LAB color space. It can be seen that in the a and b channels, the two images are very similar, even though in the RGB version the variations in lighting make them appear different.

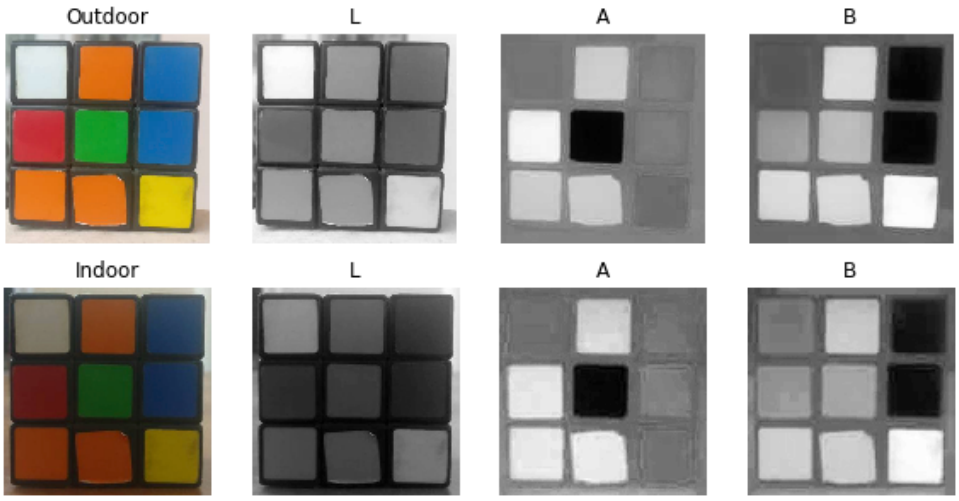

Figure 2.34: LAB Colorspace representation of a Rubik's cube in varied lighting [48].

Histogram Equalization

When cameras record image data, they do so using parameters that define shutter and lens aperture settings, which determine how much light the sensor is allowed to receive. There is a constant trade off between capturing too much and too little light. Dark regions in an image (e.g shadows) require a longer exposure time, and lighter areas need shorter exposure time to avoid over-saturated regions (e.g "whiteouts"). Oftentimes, it is difficult, or even impossible, to accommodate both of these conditions. Additionally, once image data has been recorded there is nothing that can be done to change these parameters. However, it is possible to leverage information that is recorded to lessen the effects of parameters that were not accurately tuned. This process is known as Histogram Equalization and considers the distribution of pixel intensity values across an entire image [50]. An example of a raw image can be seen below in Figure 2.35, along with its accompanying intensity distribution. The process of Histogram Equalization involves utilizing the cumulative distribution function, which essentially spreads out the original intensity distribution. The histogram equalization algorithm was applied to the raw image, and can be seen in Figure 2.35 along with the new intensity distribution. The first image is underexposed, leaving the absence of

detail in certain locations. The second image is after the application of Histogram Equalization. There image is a lot easier to make out. Note that some detail is lost on the leftmost flag, as it is now somewhat overexposed.

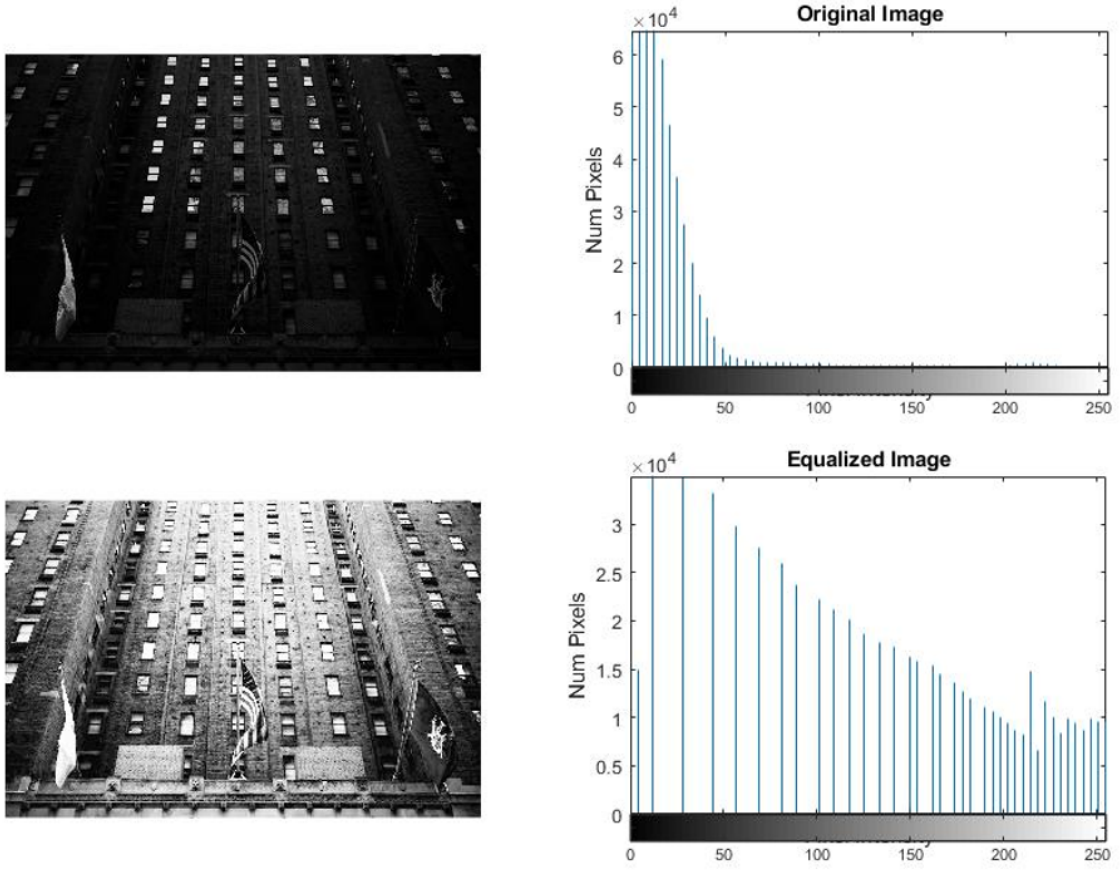

Figure 2.35: Histogram Equalization Applied to an Underexposed Image [82].

To combat this, a technique has been developed that builds upon the histogram equalization method outlined above. The method is known as contrast-limited adaptive histogram equalization (CLAHE) [49]. In this method, the first step is to divide the image into a number of grid elements, essentially making a quantity of individual images. At this point, histogram equalization is performed on each individual grid element. The result is an image that retains detail while equalizing the image intensities. This comes at a cost of additional

computational requirements, especially at smaller grid sizes. The results of CLAHE can be seen in the Figure 2.36.The Left image is one that was equalized using histogram equalization over the whole image. The right image uses CLAHE and it can be seen that more detail is preserved throughout the whole image.

Figure 2.36: Histogram Equalization vs. CLAHE Method [82].

2.3.2 Basic Computer Vision Algorithms

This section details computer vision algorithms that can be thought of as building blocks as they provide inputs to other algorithms designed for more specific applications. These more specific algorithms will be detailed later on, but generic algorithms are described here for later reference.

Edge Detection

Within the field of computer vision, edges are one of the most necessary features contained in images. Edges provide context of the structure of the content within images, and are often used as a preprocessing step in computer vision algorithms. An edge can be thought of as a line in which there are abrupt intensity changes between regions in images. There are a few different edge detectors, one of the most popular is the Canny Edge Detector which is

built upon earlier Laplace filters [51]. The biggest improvement over earlier filters is that the Canny Edge Detector organizes edge-candidates into contours. An example of the Canny Edge Detector algorithm's output is shown applied to the truck image in Figure 2.37. The Canny Edge Detector is easily accessible through OpenCV, which has a built in function to handle the operation. The function accepts a single channel image, and parameters which define the upper and lower threshold on the gradient between pixels.

Another popular option is the Sobel Edge Detector which approximates the derivative by convolving a Gaussian kernel over an image. Similarly, OpenCV has a function for doing Sobel edge detection which also accepts a single channel image. The parameters include the precision of the output image, as well as the order of the derivative in the x and y direction. Thresholds are applied to the result to determine what constitutes an edge.

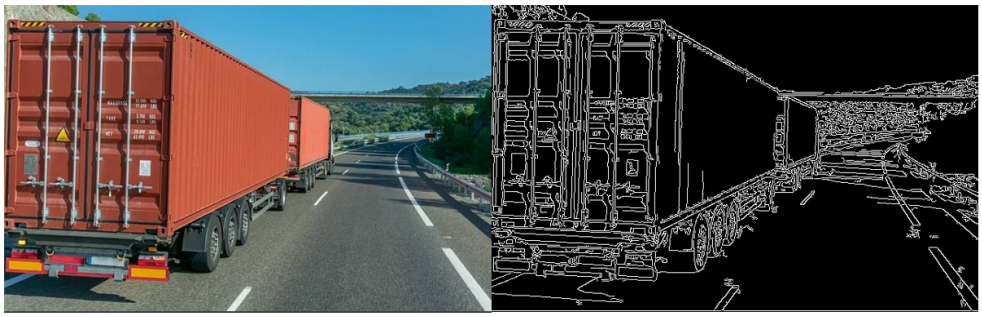

Figure 2.37: Result of Canny Edge Detector.

Hough Transform

The Hough Transform is a computer vision method for finding lines, among other shapes and forms, in an image [52]. The method was originally developed for use in physics experiments in the 1950s, and later found applications in computer vision. The method approaches the line following problem by treating every point in a binary image as though it could belong to a set of an arbitrary number of lines contained in the image. Each line can be parameterized, utilizing polar coordinates, by an angle θ and radius from origin ρ. This makes a

representation of the line in this form $\rho = x\cos(\theta) + y\sin(\theta)$. A locus of points is created in the (ρ, θ) plane which correspond to all of the lines passing through the point, of which it is possible a part of. By converting every non-zero pixel value contained in the binary image to points on the (ρ, θ) plane, and summing all of the points existing at coordinate, it becomes possible to find points in the accumulator plane which correspond to lines in the image. This is shown in the Figure 2.38 which represents the transformation of lines into the accumulator plane. Two peaks are shown in the accumulator plane, and represent the accompanying two lines in the original image. From these two points it is possible to determine the two lines' location.

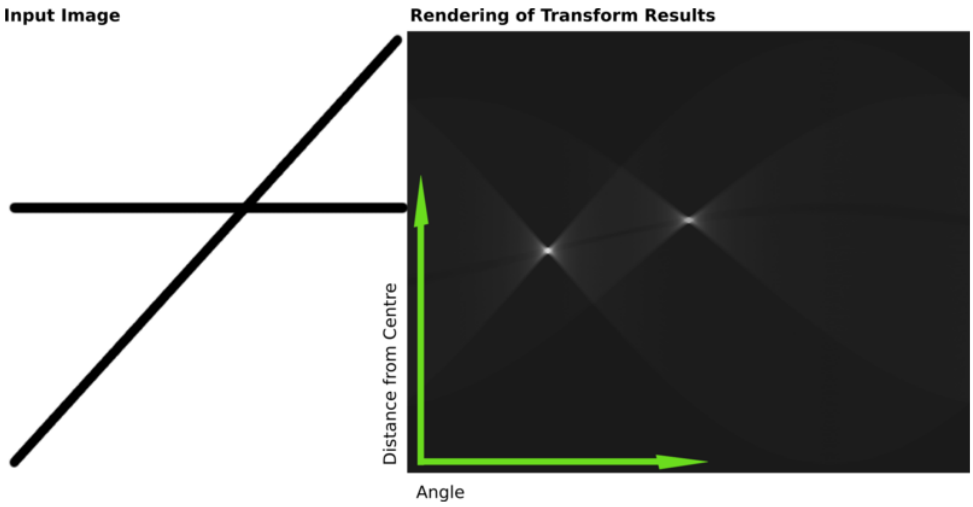

Figure 2.38: Representation of the Hough Transform and Accumulator Plane [73].

An example of the Hough Transform applied to a real image (the Canny Edge Detection image from Figure 2.37) can be seen in Figure 2.39. It can be seen that it accurately captures the edges of the trucks; however, there are additional lines captured which might not be anticipated. The way these are typically dealt with is through the use of a binary filter which uses a window of acceptable angle values to pass and fail lines which are too different from what the expected angles would be.

Figure 2.39: Probabalistic Hough Transform Function (OpenCV) applied to the canny edge detection algorithm from Figure 2.37.

Inverse Perspective Mapping

Some commercial vehicle companies have begun marketing vehicles which are able to get a 360° top-down view of its surrounding for use as a parking aid and backup aid. An example of the method shown in a commercial advertisement is shown in Figure 2.40. This application is actually done with the use of Inverse Perspective Mapping (IPM), which is sometimes referred to as "birds-eye view" [53]. This algorithm is popular for many mobile robotic applications, including autonomous vehicles. By placing a vehicle's surroundings in a top-down image view, it is possible to navigate in this plane, and even bring in other sensors which may prove useful for specific applications. The algorithm itself can be thought of as simply a re-mapping of every single pixel in an image, a function that tells each pixel where its new location is. This mapping function in the format utilized by OpenCV can be seen in Equation (2.13):

$$dst(x,y) = src(\frac{\Gamma_{00}x + \Gamma_{01}y + \Gamma_{02}}{\Gamma_{20}x + \Gamma_{21}y + \Gamma_{22}}, \frac{\Gamma_{10}x + \Gamma_{11}y + \Gamma_{12}}{\Gamma_{20}x + \Gamma_{21}y + \Gamma_{22}}) \tag{2.13}$$

Here, *dst* is the resulting top-down image, and *src* is the original image. Γ is referred to as the Transformation Matrix.The transformation matrix is used to transform each pixel stored at an x, y coordinate in the original image, to a new x,y coordinate in the inverted image. The transformation matrix is a function of four other matrices: the Projection Matrix (A_1),

Figure 2.40: Application of IPM in a commercial vehicle's user interface.

the Rotation Matrix (R), the Translation Matrix (T), and the Intrinsic Matrix (K). The projection matrix A is simply a function of the width and height of the image. This is seen below:

$$A_1 = \begin{bmatrix} 1 & 0 & -w/2 \\ 0 & 1 & -h/2 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

The Rotation Matrix is given by:

$$R = R_x R_y R_z \tag{2.15}$$

where:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & -\cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

$$R_y = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.17}$$

$$R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

In Equation(2.14-2.18), α, β, and γ are tuning parameters that control the rotation of the image. These can be determined analytically by placing a checkerboard on the ground in front of the vehicle and allowing an algorithm to approximate the appropriate values to make all of the elements on the checkerboard perfectly square. Alternatively, it is possible to hand tune the values in order to get an approximate value for the parameters.

The Translation Matrix T is defined as follows:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

where, d is a parameter that accounts for distance, and scales the image appropriately. The final matrix, is the Intrinsic Matrix (K) which should be identical to the K obtained from calibrating the camera being used. It can also be expressed utilizing the focal length of the camera, as well as the width and height of the image.

$$K = \begin{bmatrix} f & 0 & w/2 & 0 \\ 0 & f & h/2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.20}$$

The results of IPM applied to a driving scene can be seen below in Figure 2.41. Note that the road appears top-down, but the lead truck gets severely stretched. This is because IPM disproportionately stretches pixels which are located further away from the camera.

Figure 2.41: The original image (Left) and resulting IPM image (Right).

2.3.3 Keypoints and Descriptors (Features)

Many modern computer vision algorithms rely on the use of keypoints, also sometimes referred to as features [55]. The key idea is to represent an image of an object or scene in a form that will be similar, in other images of the same object or scene. Feature detection is an excellent method of doing this. A keypoint is defined as encoding information from a unique cluster of pixels that is easily recognizable. The details defining a keypoint are summarized in what is called a descriptor, which mathematically represents information which is useful to recognize the same group of pixels in another image. There are many keypoint types which are commonly used. A select few are listed below.

Harris Corner- The Harris Corner can be thought of the prototypical keypoint. In the derivation, corners can be defined in an image where the autocorrelation matrix has two large eigenvalues. It was later found by Shi and Tomasi that good corners were those that the smaller of the two eigenvalues was greater than a minimum threshold.

FAST- The Fast Feature Detector utilizes a comparison between point P and a series of points encircling it, as seen in Figure 2.42. The brightness of these pixels are compared against P and are placed into 3 categories: brighter than P, darker than P, or comparable to P. The algorithm considers points on the circumference which form an arc of continuous brightness. These are used to define a unique cluster of pixels.

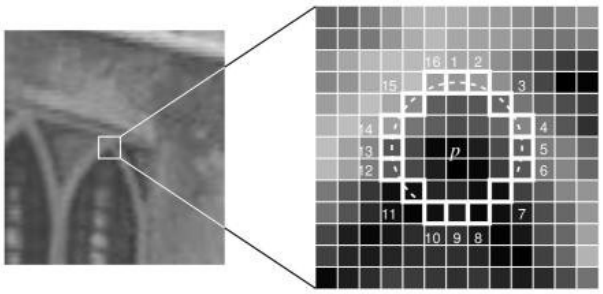

Figure 2.42: Representation of the FAST Feature Detection Algorithm [55].

2.3.4 Optical Flow

The Optical Flow problem can be defined as attempting to find features in one image which have moved to another location in a consecutive frame [55]. This has applications in motion estimation, both of elements in a scene, and also of the camera itself. One of the most popular optical flow algorithms is known as the Lucas-Kanade algorithm [84]. Originally designed in 1981, for use in determining dense optical flow (every pixel in an image), the algorithm has found a common role in modern computer vision as the means to compute sparse optical flow. The algorithm relies only on local information, contained to a small region surrounding a specific feature. One drawback to this is that large feature movement between frames can move features outside of the region. This has led to the

creation of the "Pyramidal" LK Algorithm, which functions by tracking at a highest level of an image pyramid, and works downward to account for larger motions [83]. There are three fundamental assumptions made by the Lucas-Kanade algorithm. The first assumption is that there is a brightness consistency between frames in which the object's appearance does not change. The second assumption requires that the camera update rate be fast enough to make sure that feature motion from frame to frame remains small. The final assumption is referred to as spatial coherence. This essentially means that neighboring features in an image, belong to the same surface or body, and will therefore have similar motion.

2.3.5 Common Vision Algorithms for Autonomous Vehicle Applications

Visual Odometry

For autonomous vehicles, the undisputed king of localization is a GNS/INS fusion based solution. The Global Positioning System (GPS) solution can provide a high fidelity position fix, when GPS signals are available. An Intertial Navigation System (INS) solution, which uses an inertial measurement unit (IMU), can also provide a high fidelity position fix by integrating linear accelerations and rotational velocities. One issue with INS is that the position solution error drifts in time, gradually increasing the longer it has been running. This attribute makes a GNS/INS solution an ideal combination. The GPS can provide a position fix when satellites are visible, allowing for the INS to reset the drift error with an accurate location. While this solution is tried and tested, there are a few problems associated with it. The first being that GPS might not always be available, which has spurred autonomous vehicle research to investigate methods that can provide a localization solution when GPS is not available. Another issue is that high quality IMUs are expensive, and cheaper IMUs might create bottlenecks with position solution accuracy. Outside of autonomous road vehicles, indoor robotics must also consider solutions which do not use GPS, because of attenuated signals due to the surrounding structure. These reasons have caused researchers to investigate a method known as Visual Odometry, a method for localization utilizing a camera,

by tracking features over consecutive frames [56]. The basic visual odometry algorithm for monocular cameras can be seen in Algorithm 1. Additionally, an example of the visual odometry algorithm in practice can be seen in Figure 2.43. It shows not only the features detected and utilized for the odometry measurment, but also the plot of the path taken by the vehicle.

Algorithm 1 Prototypical Monocular Visual Odometry Algorithm

1: Capture frame I_k
2: Determine features in I_k
3: Track features in image pair I_k and I_{k-1}
4: Compute Essential Matrix
5: Decompose Essential Matrix into R_k and t_k, creating T_k
6: Rescale t_k ▷ Scaled from other sensor or velocity estimate
7: Concatentate transformation by computing $C_k = C_{k-1}T_k$
8: Return to (1)

While visual odometry can be an effective localization solution, it does have its own associated issues. One of the largest problems occurs when features are detected on objects or scenery which can move independently of the ego-vehicle (i.e. other vehicles, pedestrians, etc.). These features create issues when trying to compute motion. Another major issue is that visual odometry requires a minimum number of features in order to function properly. This issue can present itself in low-light situations in which features simply are not perceivable, or can also be due to environments lacking features altogether. This is especially an issue for indoor mobile robots (i.e. blank walls). It also should be noted that there are some other variants of the visual odometry algorithm outlined in Algorithm 1. This includes stereoscopic camera versions, which can estimate velocity, removing the need for an external velocity measurement [86], and also includes a version that incorporates LiDAR [85].

Lane Following

One of the most common ways that computer vision is used in autonomous vehicles is lane detection and following. As the name implies, the problem involves detecting lane

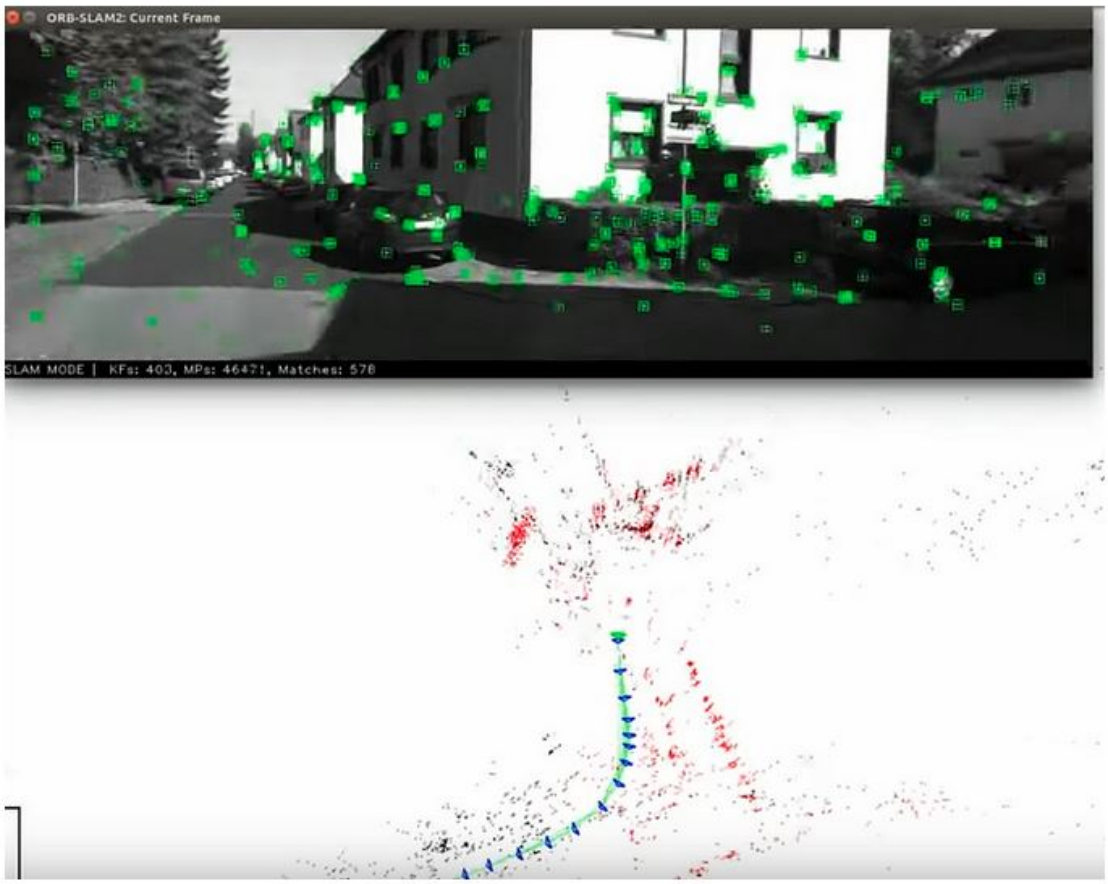

Figure 2.43: Example of Visual Odometry Algorithm applied to the KITTI data set, and used as part of an Orb Slam scheme [74].

markings (or in some cases a road edge), and uses them to determine an appropriate lateral offset that can be used by a lateral controller. The lane following algorithm works in tandem with a localization solution in order to perform high level navigation while local control is dictated by lane following along with other onboard sensors. There are a few different approaches to lane detection and all perform the same application, but the methods differ in various ways resulting in different qualities of results.

Simple Hough Transform

Perhaps the simplest method for detecting lane lines involves utilizing the Hough Transform to detect the lines directly as outlined in Algorithm 2 [58]. The caveat is that the lane lines must be straight, as issues arise when roads with a high radius of curvature are encountered. The method applied an image is seen in Figure 2.44 and shows the output at each respective step in the algorithm. The method first uses an edge detector to find edges contained in the image, typically the canny edge detector. Following the detection, a region of interest is selected immediately in front of the ego vehicles, with the assumption that the lane lines will be in that space. The Hough Transform is then used to detect the lines in this region. In order to remove erroneous lines that were detected, the angles of the lines are calculated, with lines deviating too much from expected ranges being removed.

Algorithm 2 Simple Hough Transform Lane Following Algorithm

1: Capture frame
2: Canny Edge Detection
3: Region Of Interest Determination
4: Hough Transform to find lines in ROI
5: Accept Lines with $30^\circ < \theta_r < 60^\circ$ or $115^\circ < \theta_l < 145^\circ$

Prototypical Lane Following Algorithm

There are issues with using the above lane following algorithm, namely the fact that the Hough Transform methodology only detects straight lines. This causes issues when

Figure 2.44: Original Image (left), Canny Edge Image (center), Detected Lane Lines (right). The detected lane lines can be seen in blue, and the region of interest can be seen in green.

navigating turns. To combat this, a more sophisticated lane following method has been created, and is the basis for most lane detection and following algorithms such as the example discussed in [59]. Slight variations exist, but most are made up of the fundamental steps given in Algorithm 3, and is tuned for a specific vehicle or environment. The steps of the algorithm are shown visually in Figure 2.45. First, the input image is converted into a binary threshold image. This can be done in many ways, including filtering based off of color or intensity, or other image filtering techniques. The outcome of these processes is an binary threshold image that contains the lane lines, as well as some potential pixel clusters which don't belong to the lane line (noise). This noise can be removed through additional filtering techniques, and by considering a region of interest in which the lane lines will exist. The region of interest can be a simple rectangle or a trapezoid. At this point, the only thing left in the image should be lane lines. From here, the image is inverted using IPM. A sliding box algorithm is then applied to turn the continuous lane lines into a series of discrete points. The sliding box method is initialized by a region at the base which contains the lane line. The centroid of the pixels in the box is determined and is assumed to exist on the lane. A new box is then "stacked" on top of the previous box, and its initial position is a function of the centroid of the previous box. The "sliding" in the name is due to the sliding action of the boxes to fit the curvature of the lane lines. A curve fitting algorithm is then used to model the lane lines as a curve. An example output of the algorithm with its corresponding input is provided in Figure 2.46.

Algorithm 3 Prototypical Lane Following Algorithm

1: Capture frame
2: Generate Binary Threshold Image
3: Region of Interest Filtering
4: Inverse Perspective Mapping
5: Sliding Box Filter
6: Curve Fitting

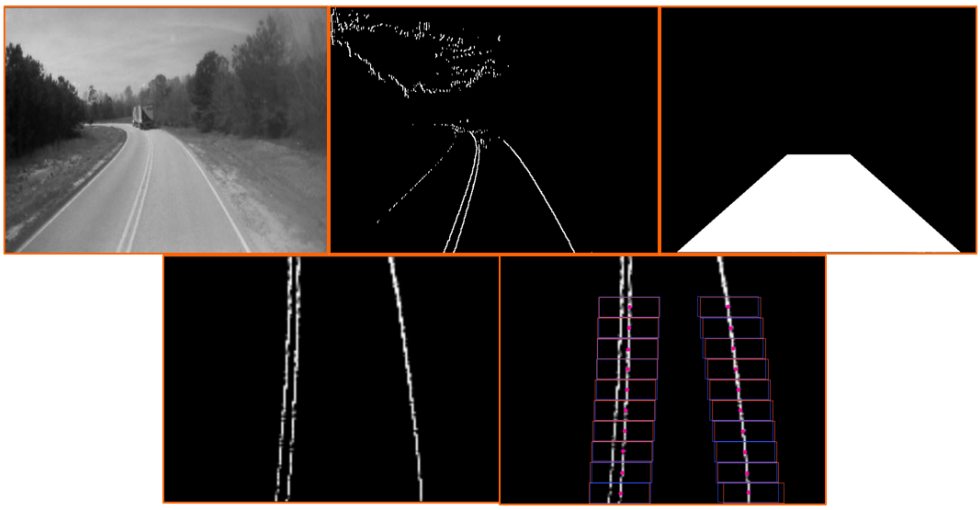

Figure 2.45: Original Image (left-top), Binary Threshold Image (center-top), Trapezoidal Mask (right-top), IPM Result (left-bottom), Sliding Box Detector (right-bottom).

Neural Network Based Approaches

Another common method for lane detection is utilizing a convolutional neural network (CNN) to detect lane lines. The method, as with all CNNs, is to annotate a large quantity of data, with the hope that a generic self-correcting model will be able to fit the data. In LaneNet , a research project that utilizes a CNN to detect lane markings, 5000 images were annotated for lane markings [60]. These images are still inverted using IPM, as was the more traditional lane detection and modeling method. However, LaneNet is trained with the objective of avoiding false positives when there are vehicles cluttering the roadway.

Figure 2.46: Example Output of Lane Following Algorithm outlined in Algorithm 3.

Example outputs of LaneNet can be seen in Figure 2.47. The network itself consists of two individual neural networks. The first is the lane edge proposal network, which serves to generate hypothesis lane markings. The hypothesis lane markings are then utilized by the lane line localization network.

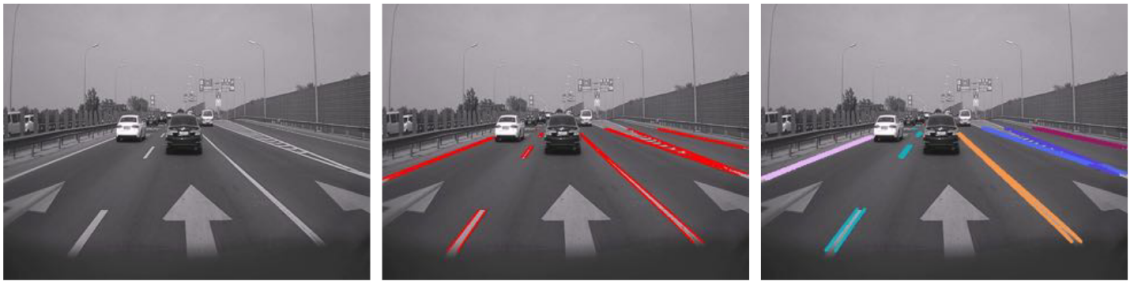

Figure 2.47: Outputs of LaneNet. Original Image (left), Proposed Lane Lines (center), Accepted Lane Lines (right) [60].

Simple Road Edge Detection

There are certain situations in which lane markings might not be visible, or even present at all. In this case, road edge detection and following is used to estimate the road edge position instead. This functions by detecting either the color or texture of the drivable road surface. An example of this type of algorithm, which was designed to function on dirt roads, is outlined below in Algorithm 4. The algorithm is based on [54], which used the method on paved unmarked suburban roads. The algorithm first defines a region of interest immediately in front of the vehicle, and determines the average pixel intensity of the pixels

within the region of interest. An assumption is made that the drivable road surface will have an intensity comparable to this average. Using IPM the image is inverted, and filtered based on this average intensity leaving pixels which are the drivable road surface. The edges of the filtered output are then the detected road edges.

Algorithm 4 Road Edge Detection Algorithm for Dirt Road

1: Determine Region of Interest
2: Calculate Average Pixel Intensity in ROI
3: Invert Image Using IPM
4: Filter Inverted Image for Pixels with Intensity Similar to average intensity in ROI
5: Bound the Region of Pixels with Similar Intensity (this is the drivable road surface)

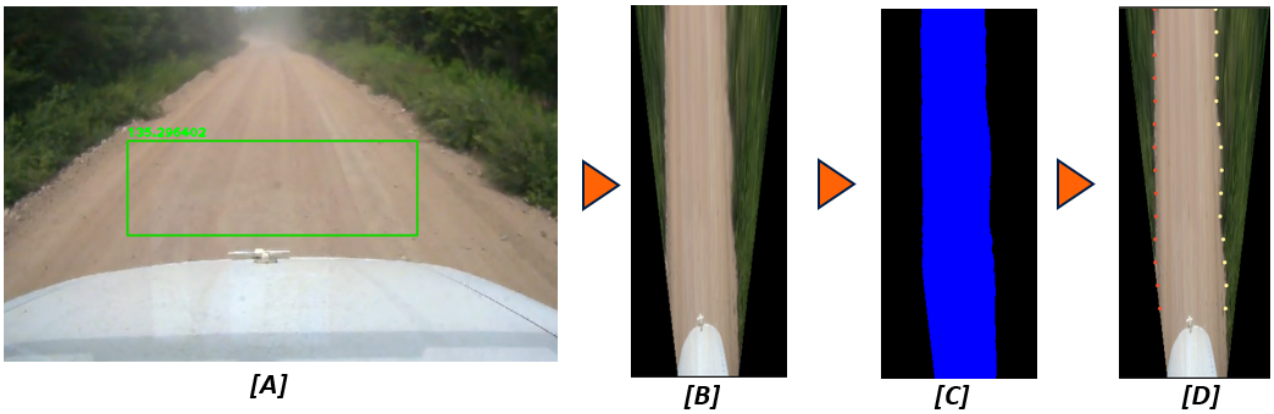

Figure 2.48: [**A**] Input Image and ROI with calculated average intensity, [**B**] Inverted Image, [**C**] Filtered Inverted Image, [**D**] Bounded Road Edges from [C].

2.3.6 Object Detection and Tracking

Object Detection is a class of problem that concerns itself with annotating specific objects within an image frame. Object Tracking, in comparison, involves tracking an already annotated object over consecutive frames, essentially re-recognizing the same object in a new frame. These two problems have formed a collective problem in the form of Object Detection and Tracking, which attempts to solve both problems simultaneously. An example of a very simple version of an object detection and tracking algorithm might be created by considering

the problem of detecting and tracking a tennis ball as seen in Figure 2.49. The input image would first be converted to the HSV colorspace, and then filtering the image based on color leaving only pixels that were within a certain range around the yellow color of a tennis ball. The largest cluster of pixels could then be assumed to be tennis ball. The problem with the method is that it is not very robust, any object with a similar color could be detected and be confused with the target object. This is also true for changes in lighting, which might cause the object to no longer meet detection criteria. It also would not consider more than one tennis ball.

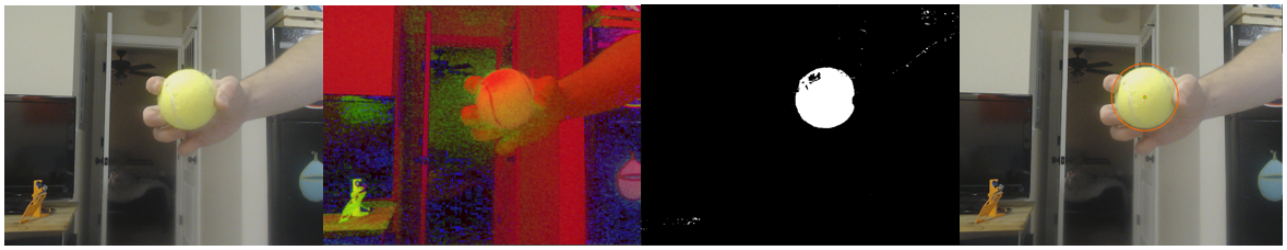

Figure 2.49: Simple HSV based object detector. From left to right: Original Image, HSV Image, Filtered Image (Binary Threshold), Largest Contour (with bounds and centroid annotated).

Blob Detector

Blob detectors are a more advanced version of the very simple detector described above [55]. Given a binary threshold image, Blob detectors sort out the groupings of pixels based on their geometric properties. These properties include the area of the "blobs", along with things like circularity (how close a blob is to a circle) as well as inertia (how elongated a blob is). Blob detectors work best in situations where the tracked objects are on a simple background with constant lighting, such as industrial robots. OpenCV has a class implementation of the blob detector called the SimpleBlobDetector. The implementation allows for a custom blob detector by passing parameters that describe blobs. A visual represnentation of these parameters is provided in Figure 2.50. It should be noted that any combination of

the parameters could be used, even just one of them. SimpleBlobDetector even provides for a thresholding parameter to convert grayscale images into binary images. The parameters utilized by the SimpleBlobDetector are defined:

1. **Thresold** Minimum and Maximum Pixel Intensity (0 - 255) → This step creates blobs
2. **Area** - Minimum Area to be considered a blob (area in pixels)
3. **Circularity** - How Close a Blob is to a Circle (0 - 1) → circularity of circle is 1, and square is 0.875
4. **Inertia** - Measure of Elongation of Blob, opposite of mechanical moment of inertia (Ranges from 0 to 1) → Inertia of Line is 0 while inertia of Circle is 1
5. **Convexity** - Defined by Area of Blob over Area of the Convex Hull-Tightest Convex Shape to Completely Bound the Blob. (0 - 1)

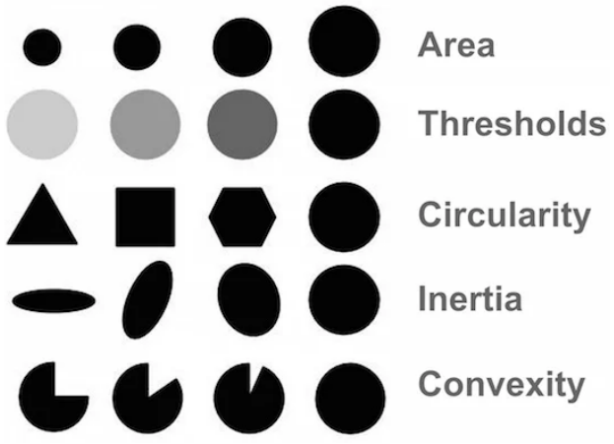

Figure 2.50: Visual representation of parameters utilized by SimpleBlobDetector [87].

Modern Neural Network Methods

In computer vision, object detection and tracking is focused on recognizing and annotating instances of objects in the image frame, typically with bounding boxes. You Only

Look Once (YOLO) is a real-time object detection algorithm which utilizes a convolutional neural network to classify objects [8]. The working principles of the YOLO algorithm can be seen in Figure 2.51. First, the algorithm breaks the image into an SxS grid, and hypothesis bounding boxes are generated while simultaneous grid probability determined for each cell. Using the hypothesis bounding boxes and probabilities, the final detected bounding boxes are provided. Each bounding box is accompanied by 5 states: width of bounding box, height of bounding box, x-coordinate in the image frame, y-coordinate in the image frame, and class label. Using YOLO has many advantages, including how fast the algorithm can run. Additionally, because YOLO analyzes the input image in its entirety, the algorithm can leverage contextual information in its class prediction[8]. Some updates have been provided to YOLO in 2018 with the release of YOLOv3 [75], which is used in this thesis.

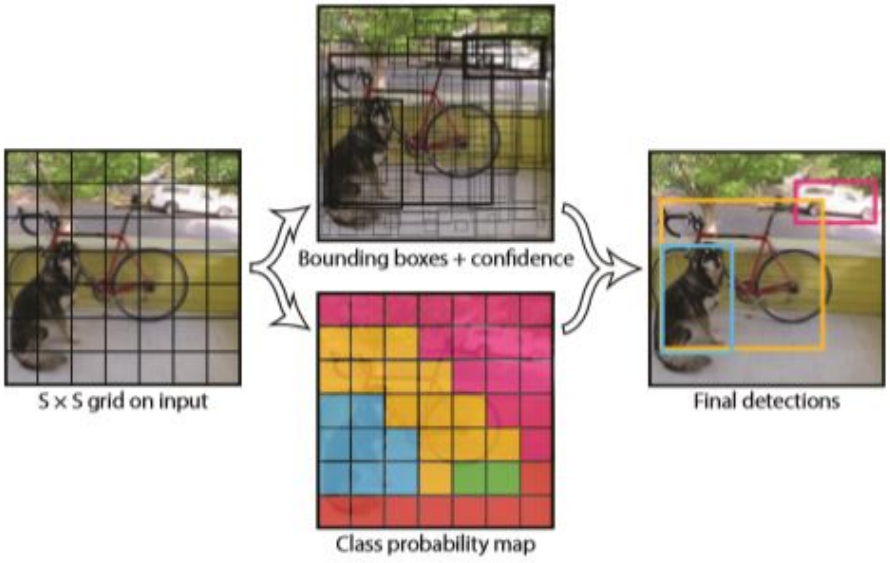

Figure 2.51: YOLO Algorithm Steps [8].

2.4 Conclusion

This chapter serves to provide the necessary background for understanding the work described in the remainder of the thesis. Topics covered have included how a traditional GPS/INS platooning platform operates in detail, providing a framework for how a camera

might improve the system by providing a relative position vector without GPS. A historical perspective of the camera, as well as how a camera is modeled is also discussed. This is necessary for understanding how camera based detections of a lead vehicle can be used to determine an RPV. Additionally, methodologies which could be used to detect objects using a camera was provided to show context of how the lead vehicle is detected in the following chapters. Lane detection and following methods, as well as visual odometry were provided to aid in future development of the methodologies. The next chapters detail how these techniques were utilized to generate RPVs for heavy duty trucks, as well as consumer sedans.

Chapter 3

YOLO Based Relative Position Vector Generation for Vechicle Platooning

3.0.1 YOLO Implementations

Four implementations of YOLOv3 were trained using the Darknet Framework [8]. The networks were trained with road training data collected on various road conditions. These implementations include YOLOv3 and Tiny YOLO. Two versions of each exist, one for RGB data, and one for IR data. 400 RGB images, and 400 IR images were used for training. As part of a dataset collection three laps were performed on roads near the National Center for Asphalt Technology (NCAT) testing facility. The images used for training the YOLO implementations were from the first two laps of the data collection run, with the third lap data being reserved for evaluation. The default YOLO parameters were mostly retained, with the exception of those associated with the number of classes [61]. The training time for each YOLO implementation took approximately two hours on a computer featuring an NVIDA RTX 3070. Robustness analysis was performed to evaluate the accuracy of the implementations, the results of which are provided in Section 3.2.1. The trained weights were then incorporated into ROS nodes to handle time syncing with truth data. An output of the trained YOLO algorithm can be seen in Figure 3.1. Additionally, the aspect ratio of the bounding box was considered to reject objects that exceed a threshold as seen below in Equation (3.1):

$$0.8 < \frac{W_y}{H_y} < 1.2 \tag{3.1}$$

Figure 3.1: Example of Bounding Box Provided by YOLO

3.0.2 Pinhole Model

A modified pinhole model was considered as one of the methods for range determination. Equation (3.2) shows the pinhole model used, and Figure 3.2 shows a graphical representation of the model:

$$r = F_c l \frac{y}{y'} + C_y = L \frac{1}{y'} + C_y \tag{3.2}$$

Range is defined as r, with F_c being the focal length of the camera. y is the actual height of the truck in meters, and y' is the height of the truck in the image plane. l has the units of pixels/meters and is a pixel conversion parameter that handles the conversion of pixels to meters.F_c, y, and l are combined into one term L to simplify the model tuning process.Similarly, C_y is a parameter that handles the translation between the image plane, and the real world. All tuning parameters are available in Table A.1 in the appendix of this thesis. It should be noted that instead of using the classical pinhole model, a custom calibration curve could be created which models the range as a polynomial as a function exact height or area of the object in the image frame. Figure 3.3 provides an example of this type of calibration curve which related the area of the detection of the lead truck's bounding

box to the DRTK range truth. This method was not pursued in this thesis because it is not the traditional approach taken for monocular ranging, and the classical pinhole model was used instead.

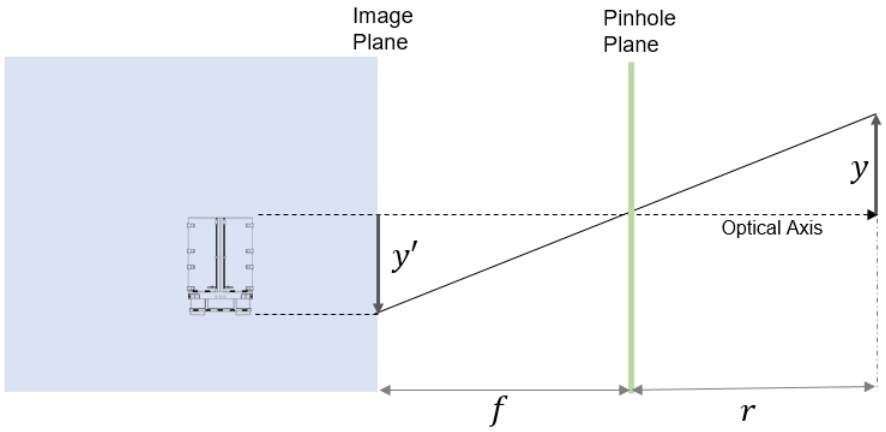

Figure 3.2: Pinhole Camera Model.

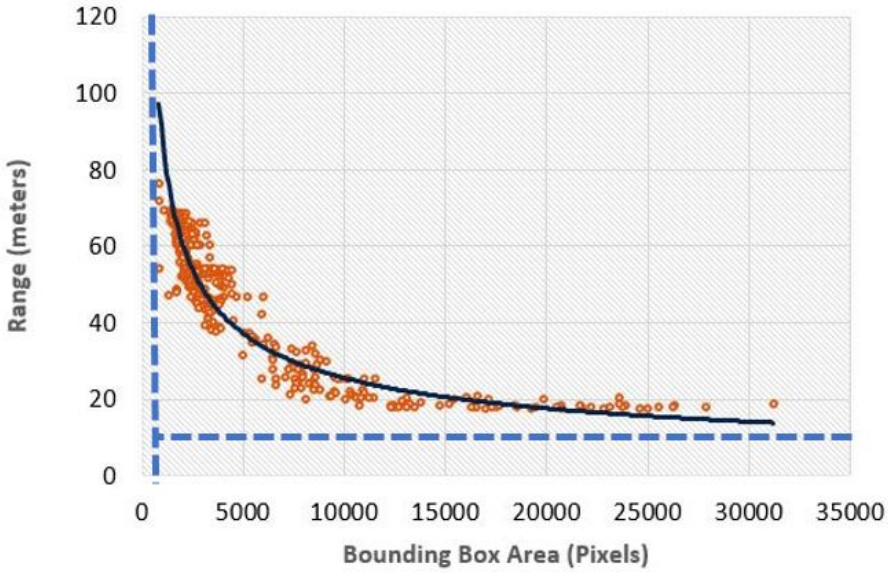

Figure 3.3: Calibration Curve Generated Using Bounding Box Area and Truth Range.

3.0.3 Virtual Horizon Model

Park et al. proposed a method for determining monocular range to a vehicle [26]. The proposed method involved computing a virtual horizon, and then computing the difference

between the bottom of a vehicle and the virtual horizon. This can be seen represented in the Figure 3.4 with the model being presented in Equations (3.3-3.4).

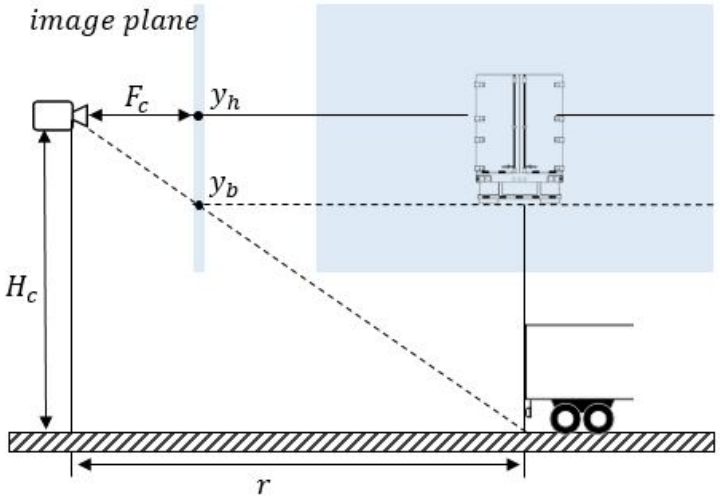

Figure 3.4: Virtual Horizon Model from [26].

Mathematically, this is represented as:

$$r = \frac{F_c H_c}{y_b - y_h} * C_{VH} \tag{3.3}$$

where F_c is the focal length of the camera, H_c is the height of the camera, y_c is the y-coordinate of the bottom of the bounding box, and y_h is the virtual horizon line's y-coordinate. C_{VH} is used as a scaling term to tune the model. The term y_h is estimated through the use of Equation (3.4):

$$y_h = y_b - H_c \frac{w_a}{W_a} \tag{3.4}$$

Here, w_a is the width of the vehicle in the image plane, and W_a is the actual width of the vehicle in units of length. All parameters used for the virtual horizon model are available in Table A.2 in the appendix. It should be noted that the method from [26] has been improved upon to estimate parameters such as the camera height, and also leverages detected lanes to avoid the need of width of the lead vehicle in [27]. The improved method is not explored in

this thesis because the width of the lead vehicle is known, and also it is desired to measure range on roads with and without lane lines. For both ranging methods, range measurements were filtered using a moving average filter.

3.0.4 Bearing Determination

Using the range to the lead vehicle it is possible to determine the bearing to the lead vehicle. The method used is drawn in Figure 3.5. First, the lateral error between the follower vehicle and the lead vehicle is calculated. This can be done utilizing the range measurement as well as the pinhole camera model in Equation (3.5):

$$L_e = \frac{L'_e r}{F_c k} \tag{3.5}$$

L_e is the lateral error expressed in real world units, L'_e is the lateral error in the image plane which is in pixels. R is the range calculated either through the pinhole model, or through the virtual horizon method. The parameter k is used to handle the real world units to pixel transformation and F_c is the focal length of the camera. Once the lateral error has been determined, the angular component of the RPV, Φ, can be determined by taking the inverse tangent of the lateral error and range as expressed in Equation (3.6):

$$\Phi = arctan(\frac{L_e}{r}) + C_b \tag{3.6}$$

where, a term is introduced to account for bias in the model, C_b. By combining Equation (3.5) and Equation (3.6), the result becomes:

$$\Phi = arctan(\frac{L'_e}{K}) + C_b \tag{3.7}$$

where the F_c and k terms have been combined into a singular term, K, to simplify tuning. All of the parameters used in the bearing model analysis are provided in Table A.3 in the appendix.

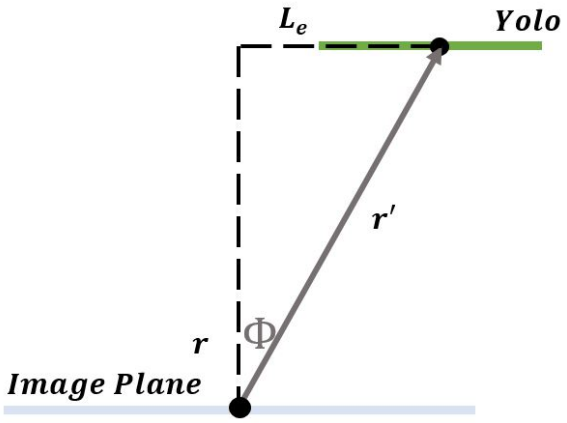

Figure 3.5: Representation of Bearing Angle.

3.1 Data Collection

Two Peterbilt 579 commercial trucks were manually driven on roads near Auburn University's NCAT facility. These trucks contained GPS receivers (Novatel Propack V3) and radar (Delphi Electronically Scanning Radar - ESR) among other sensors. Camera data was collected through the use of an RGB camera (AXIS P1214-E NET-WORK CAMERA) at 1280x720 and 30 Frames per Second (FPS). The IR data was collected using a FLIR Boson Infrared Camera which recorded at 640x480 and 30 FPS.The ground truth range is provided by the range estimator on Auburn's platform which provides the RPV between the two vehicles [30]. The route traversed is provided in Figure 3.6, along with images depicting the road conditions for each section. Multiple laps were completed as part of the data collection process. Images from the first two laps were used to train the YOLO models, and the last lap, which was driven in the opposite direction, was used for evaluation.

3.2 Results

The following sections detail a performance evaluation for the methods detailed in the preceding sections. First, a robustness analysis was performed on the YOLO implementations using an open source road vehicle dataset. Additionally the 4 different YOLO implementations were evaluated on data was collected during the third lap of the data collection

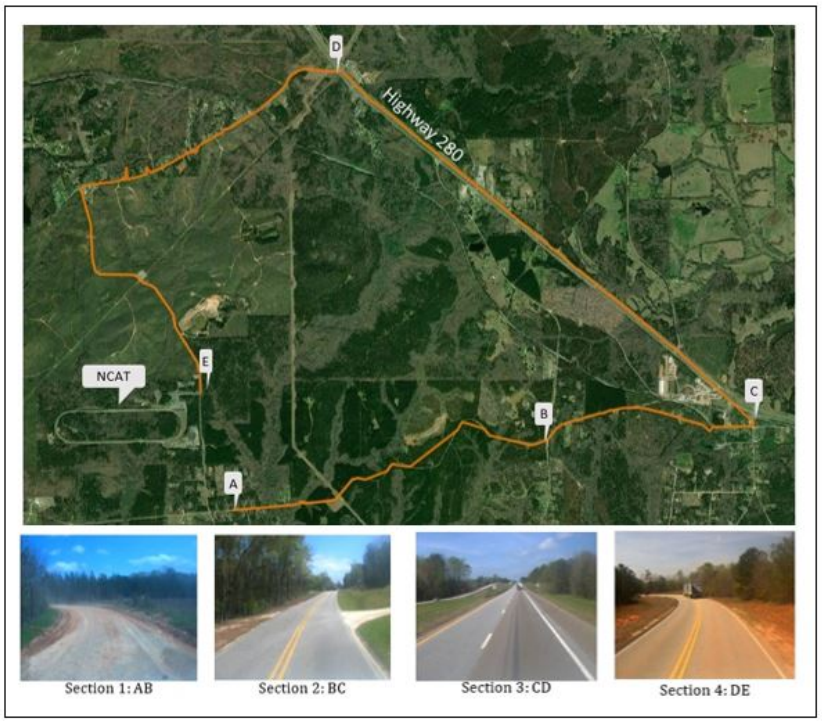

Figure 3.6: Route taken during testing [Top], and example of various route conditions [Bottom].

run, with the errors of the implementation being compared using the same model parameters. The pinhole camera ranging model and the virtual horizon model were also compared.

3.2.1 YOLO Robustness Evaluation

In order to evaluate the robustness of the YOLO implementations, an open source dataset of approximately 15,000 images of road vehicles and roadways was utilized. Only the RGB implementations were able to be tested, due to the lack of a dataset of IR images of vehicles. The exact truck that the YOLO implementations were trained on was not seen in the image dataset, meaning any detections by the YOLO implementaions would be an error. The goal of the testing was to determine the likelihood of trained networks detecting false positives on images of vehicles that are not the trained vehicle. The YOLO implementation had a false positive rate of 0.459 percent and the Tiny YOLO implementation had a false positive rate of 1.21 percent. This would be expected as Tiny YOLO has fewer weights, and

is optimized for computation speed. One of the false positives is shown in Figure 3.7, in which a window of a bus in misclassified as the rear of the truck. It should be noted that the dataset consists of images of vehicles in various poses, and might not represent a true driving scenario, which would consist of mainly rear views of vehicles.

Figure 3.7: One of the False Positives (70 images out of 15255).

3.2.2 Range and Bearing Performance

In order to test the algorithms' ability to accurately estimate range and bearing, image data from the data collection run was used for evaluation. This data came from a modified route which was longer and was run in the opposite direction. The range measurements from the Highway 280 portion of the data collection run can be seen in Figure 3.8 along with the DRTK based range measurement which is considered truth. Additionally, the bearing measurements from the same route can be seen in Figure 3.9 along with the DRTK solution's bearing measurement. It can be seen that the differing YOLO versions and camera implementations closely track the truth range and bearing from DRTK. The implementation with the highest error for this portion of the dataset was the Tiny Yolo with IR camera. It can be seen to significantly underestimate the range to the lead vehicle at longer distances (>70 meters). This behavior was also seen in the other implementations but at a lesser magnitude of error.

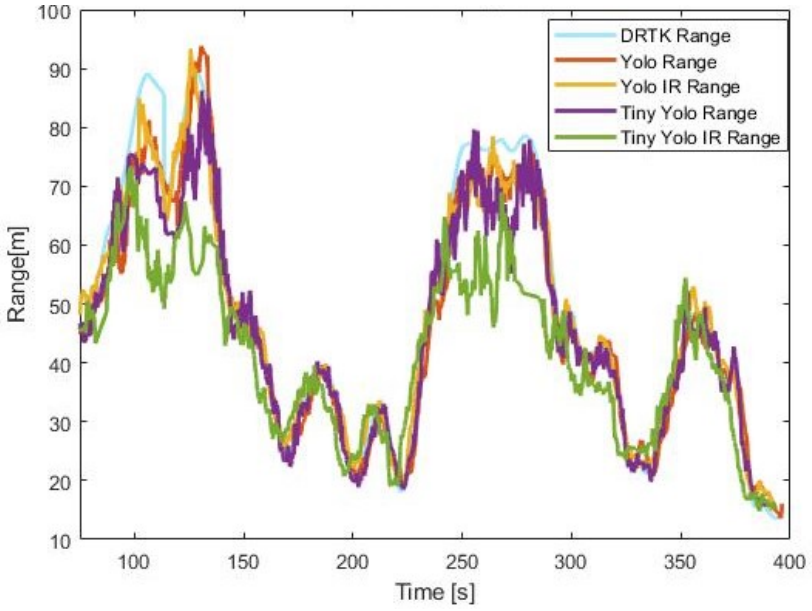

Figure 3.8: Range Results Using Pinhole Model for Section 3 of on Road Data.

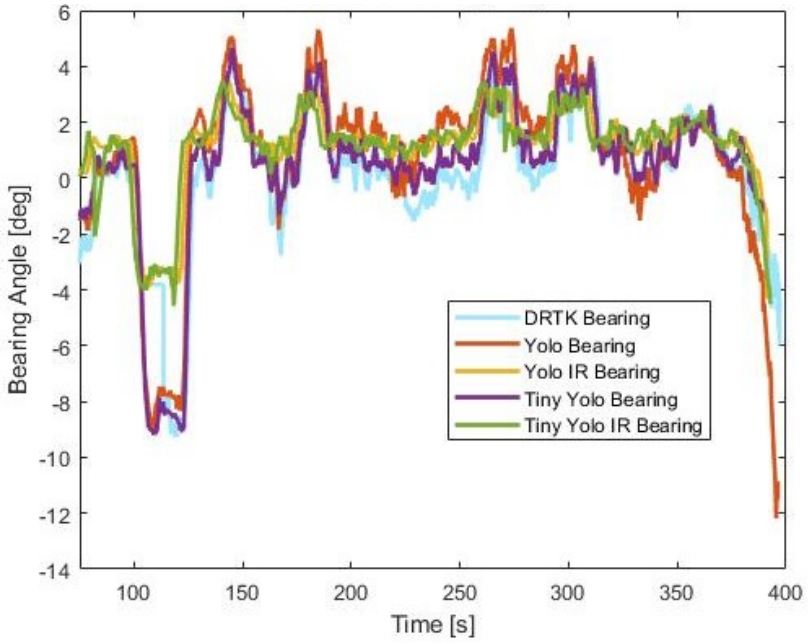

Figure 3.9: Bearing Results Using Pinhole Model for Section 3 of on Road Data.

The two range models were also compared on a portion of Highway 280 data. This can be seen in Figure 3.10. Additionally, the errors for each model can be seen in Figure 3.11 for the same portion of the run. The range models seem to perform similarly, both tracking the truth DRTK range. There is a slight underestimation of range in the pinhole camera model seen in some instances. From Figure 3.10 and Figure 3.11, it can be seen that the range error appears to correspond to the magnitude of the range. This is likely because both the pinhole model and virtual horizon model are nonlinear equations which can be approximated as linear for a limited range window. Another cause could be that the lead truck becomes indistinguishable from other elements of the image as range increases. This is especially true for IR data which has less features than RGB, including color as shown in Figure 3.12.

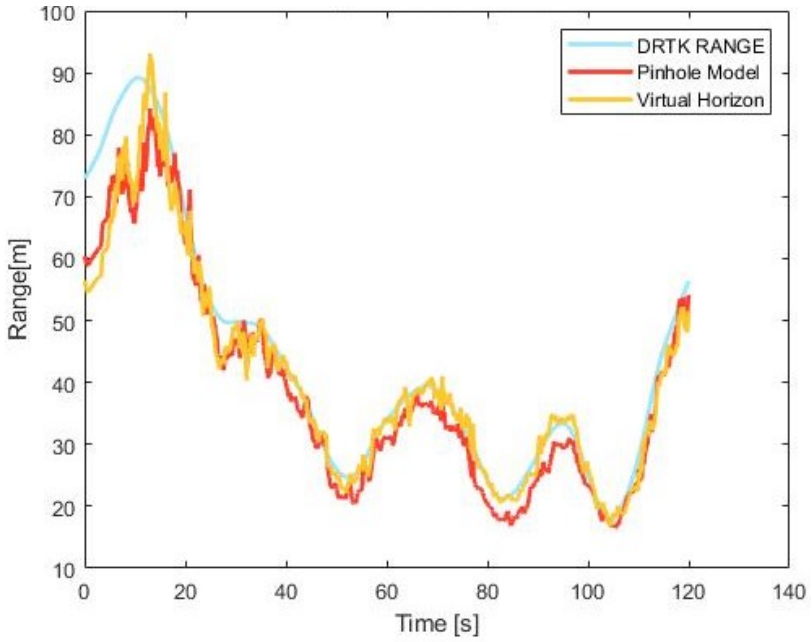

Figure 3.10: Range from Pinhole Model and Virtual Horizon Method for Section 3 of on Road Data.

To evaluate the performance of the methods, a 2 minute portion of each run was chosen to analyze. The portions were chosen for regions where the DRTK functioned properly, and also where the lead truck remained within 75 meters of the following truck while also remaining in the image frame. It is assumed that this would be reasonable to expect for an autonomous platoon which is being controlled around a desired range of less than 75

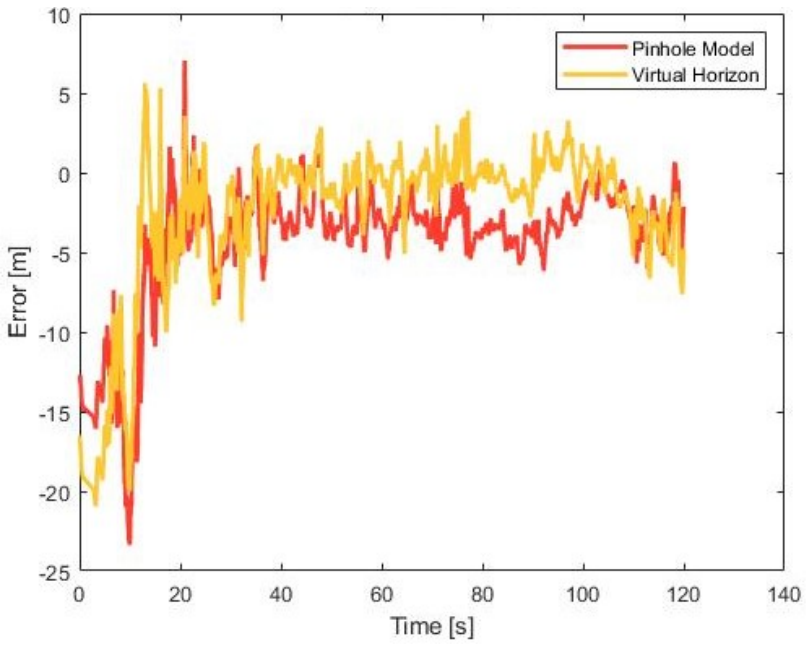

Figure 3.11: Errors associated with the two range models. It can be seen that the largest errors correspond with large ranges seen in Figure 3.10.

meters. This dataset was collected using two manually driven trucks, which occasionally exceeded this limit. However, this was consistent amongst the camera data, and allowed for the performance between them to be compared. The average mean and standard deviation for each of the methods across all runs can be seen in *Table I*. The performance metrics for each individual section of road can be found in *Table II*. The performance difference between YOLO and Tiny YOLO seems to be marginal when considering all of the runs using the same camera. The range models also seem to perform similarly, with the exception of the virtual horizon model having less mean error while having comparable standard deviation to the pinhole model. The IR camera improvement over RGB for the dirt road can be described as marginally better. Depending on the application, a user would need to decide if this improvement would be worth the additional cost associated with the IR camera.

The work presented in this chapter proves the ability to estimate range and bearing with a monocular camera for platooning applications. The tests were performed on a variety

Table I : Performance Summary

Mean,Std	All Sections						FPS	Robustness Percentage
	Pinhole Range (m)		Virtual Horizon Range (m)		Bearing Angle (deg)			
Yolo	-2.68	3.83	-1.35	3.82	0.37	0.89	7	0.459
Tiny Yolo	-3.09	3.25	-1.87	3.46	0.33	1.31	30	1.21
Yolo IR	1.45	3.42	0.45	3.11	-0.05	1.96	7	--
Tiny Yolo IR	0.21	3.51	0.44	3.40	-0.06	1.42	18	--

of road conditions, and compares IR and RGB camera data. Additionally, differing ranging algorithms were explored. After proving the merits of the monocular RPV generation methods in this chapter, a real-time implementation is explored in the next chapter.

Table II: Error for Each Method on Each Section

Mean,Std	Section 1 (Dirt Road)					
	Pinhole Range (m)		Virtual Horizon Range (m)		Bearing Angle (deg)	
Yolo	-2.88	5.08	-2.18	4.83	0.46	1.31
Tiny Yolo	-3.20	3.09	-3.28	3.87	0.48	1.91
Yolo IR	0.53	3.10	0.03	3.14	0.16	3.51
Tiny Yolo IR	-1.66	3.87	-0.41	4.07	0.04	2.26
Mean,Std	Section 2 (Rural Paved)					
	Pinhole Range (m)		Virtual Horizon Range (m)		Bearing Angle (deg)	
Yolo	-3.00	4.09	-0.87	4.12	0.18	0.38
Tiny Yolo	-3.76	3.17	-1.37	2.80	0.20	0.49
Yolo IR	1.82	4.22	-0.10	3.69	-0.17	0.76
Tiny Yolo IR	-1.15	4.22	-0.44	3.50	-0.25	0.96
Mean,Std	Section 3 (Highway 280)					
	Pinhole Range (m)		Virtual Horizon Range (m)		Bearing Angle (deg)	
Yolo	-3.23	3.87	-2.22	3.98	0.34	0.65
Tiny Yolo	-3.88	3.70	-1.93	4.00	0.36	0.77
Yolo IR	1.52	4.36	0.37	3.88	-0.11	1.78
Tiny Yolo IR	0.33	4.29	0.44	3.69	-0.26	1.01
Mean,Std	Section 4 (Rural Paved)					
	Pinhole Range (m)		Virtual Horizon Range (m)		Bearing Angle (deg)	
Yolo	-1.62	2.27	-0.12	2.35	0.52	1.22
Tiny Yolo	-1.50	3.05	-0.90	3.15	0.27	2.09
Yolo IR	1.94	1.99	1.51	1.71	-0.08	1.78
Tiny Yolo IR	3.31	1.65	2.17	2.36	0.21	1.45

Figure 3.12: Varied Visibility of Lead Truck in IR Camera Data.

Chapter 4

Improvements and Real Time Testing

Based on the initial results from Chapter 3 of this thesis, a new YOLO implementation was trained on another of our lab's vehicles, the Kia Optima shown in Figure 4.1. This was done in order to make testing easier as no CDL would be required to operate the passenger vehicles. It also allowed for more varied maneuvering in the testing environment, as well as an evaluation of the methods previously explored on a smaller vehicle. Additionally because all testing was done on a skid pad at NCAT, the wireless computer network allowed for syncing truth measurements between both vehicles through ROS distributed.

Figure 4.1: Left: 2019 Kia Optima (Leader) Right: 2019 Lincoln MKZ (Follower).

4.1 Development

The Lincoln MKZ and Kia were driven in a platooning manner (MKZ following the KIA) through a variety of maneuvers. The goal was to capture the KIA in a variety of poses

which would be reasonably expected during platooning operations. Around 400 images were collected, and hand annotated in order to train the YOLO implementation. The algorithm was trained on the trunk of the KIA because it is expected to be in view during normal platooning situations. Similar to the YOLO implementation for the heavy duty trucks, the default parameters were mostly maintained with the exception being for only training for one detection class as in [29]. A ROS implementation was created to allow YOLO to be used in real time to track the KIA. An example output image from YOLO of the KIA can be seen in Figure 4.2.

Figure 4.2: Example Output of YOLO Detection Algorithm.

4.1.1 Sub-Tracker Development

After training the TinyYoloV3 implementation, the range and bearing models were tuned using GPS data and a combination of least squares and hand tuning. The algorithms were tested in real time using a pure pursuit control scheme. This is done by utilizing the heading error as input to the controller, and attempting to drive it to zero. During initial testing, there were small lapses in the output from the vision algorithm, when YOLO would not detect the lead vehicle despite it existing in the image frame. There are a few possible

reasons for this issue such as lighting variance, or unusual poses of the lead vehicle which did not appear in the training data. To address this, a sub-tracker method was developed. The sub-tracker serves as a holdover when YOLO can not detect the lead vehicle in an image frame. Using a YOLO bounding box, a new bounding box can be propagated forward on future frames until YOLO can make a new detection. The sub-tracker uses both YOLO and a traditional object tracking algorithm. A flow chart representing the proposed method can be seen in Figure 4.3. Prior to inputting the image to the YOLO algorithm, the effects of lens distortion were removed using predetermined camera calibration parameters. Additionally, the Contrast Limited Adaptive Histogram Equalization (CLAHE) approach was used to limit potential lighting variation, and a moving average filter was introduced to mitigate noise as was done in Chapter 3. Example images of the KIA before and after the CLAHE algorithm has been applied are provided in Figure 4.4. It can be seen that details in the original image have been accentuated in the output image, which helps YOLO make more repeatable detections. The effect of the CLAHE and the Moving Average Filter can be seen in Figure 4.5. The output is less noisy, and there are less instances when YOLO cannot make a detection due to the additional detail provided by the CLAHE algorithm step.

In OpenCV, there are many open source object tracking algorithms included and they all are functionally identical. By providing the tracker an initial bounding box, it can leverage the information in the image to determine bounding boxes in future images. The basic idea for using the tracker is to have YOLO detect the lead vehicle when possible, and utilize a traditional tracking algorithm when YOLO fails to detect the lead vehicle. In either case, the most recent bounding box is used to determine range and bearing. The combination works well together because while the trackers work well, their solution tends to drift over long periods of time (5-10 seconds) and the lapses induced in YOLO are typically less than a second. A platooning scenario was performed, in which both vehicles were manually driven and raw camera data was collected. This data was analyzed in post, with an initial bounding box being manually provided to each tracker. The drifting effects were analyzed using two

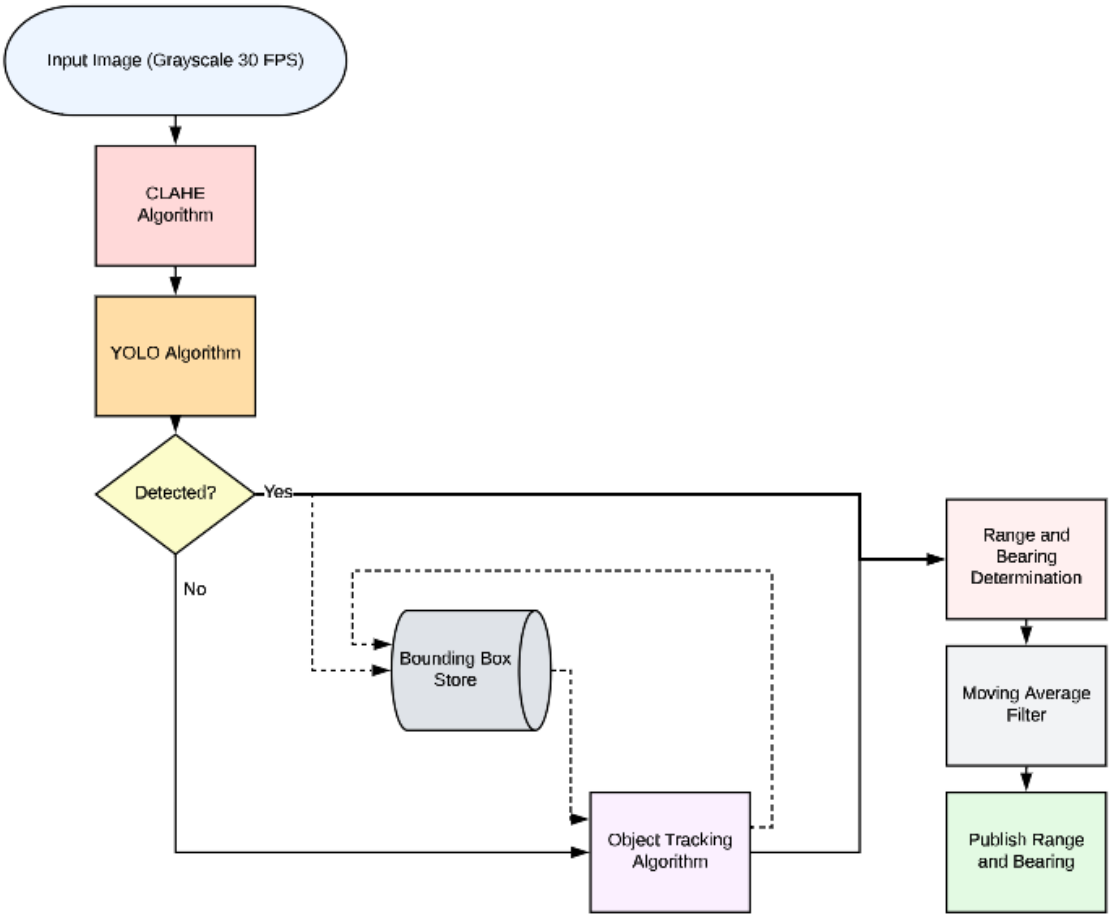

Figure 4.3: Graphical Representation of Monocular RPV Generation Algorithm.

Figure 4.4: Original Image (left), CLAHE Algorithm Applied (right).

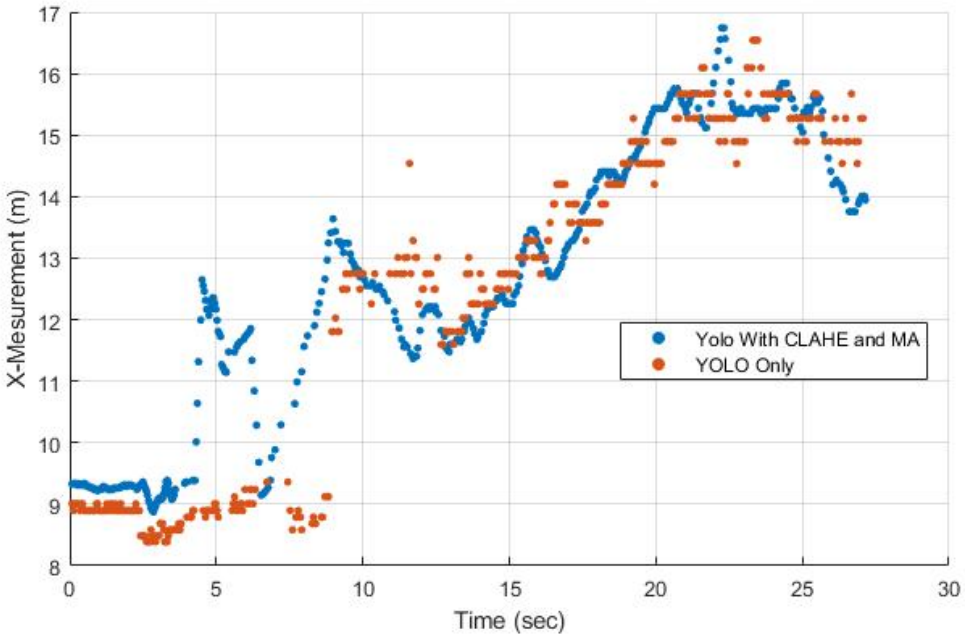

Figure 4.5: Effects of the CLAHE algorithm and moving average filter on X position measurement.

criteria using the YOLO provided bounding boxes as truth. The first criteria was height of bounding box error which is shown in Figure 4.6, and the second criteria was centroid location error which is shown in Figure 4.7. The centroid location error was determined using euclidean distance between the tracker's bounding box and the truth YOLO bounding box. In both error metrics, it can be seen that for the magnitude of error and its fluctuation grow after the first few seconds of initialization. It is expected that in the sub-tracker implementation that in this time frame, YOLO would be able to reinitialize a bounding box.

The four trackers which seemed most apt for this thesis were CSRT [77], DaSiamRPN [79], GoTurn [78], MedianFlow [76], and TLD [80] due to their dynamic bounding box sizing (some other trackers keep the bounding box the same size, and only moves in x and y). To determine which tracker would work best for tracking the lead vehicle, two manually driven data runs were performed and analyzed through post processing. In one run, the lead vehicle drove away from the follower which remained static. In the other run, which both vehicles

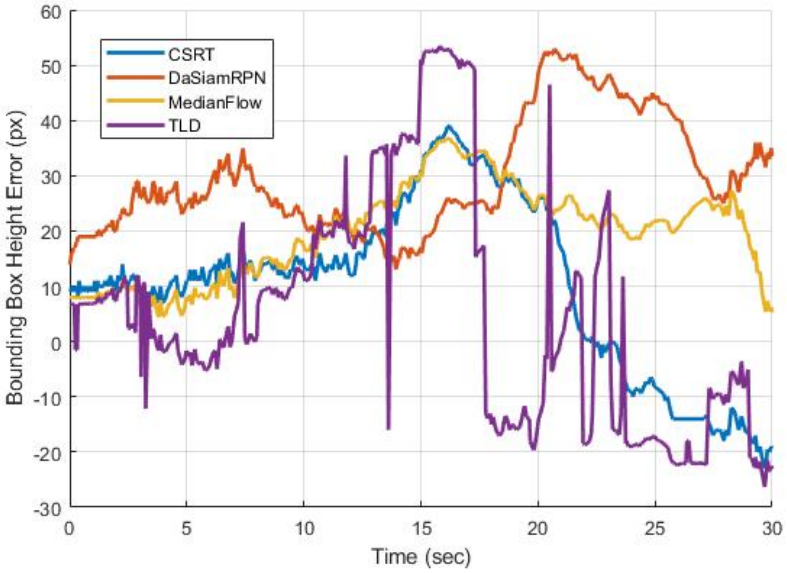

Figure 4.6: Effects of the CLAHE algorithm and moving average filter on X position measurement.

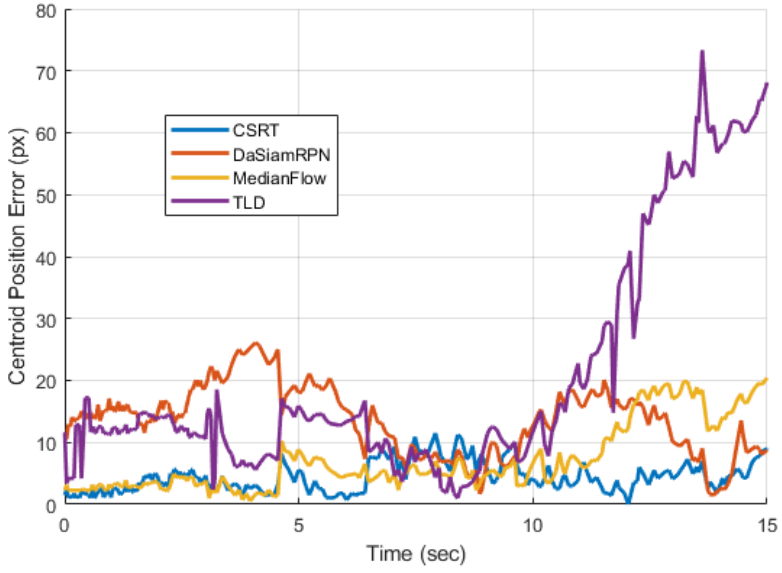

Figure 4.7: Effects of the CLAHE algorithm and moving average filter on X position measurement.

moved through an S-turn in a platooning fashion. In both runs, GPS data was used as truth for RPV comparisons. The tracker analysis was done in post by manually initializing a bounding box on the lead vehicle, and then allowing the tracker to update the bounding box through the run. The bounding boxes were used to generate RPVs which are compared against the truth GPS RPV as well as YOLO. The results from the static run are shown in Figure 4.8 where it can be seen that all four trackers perform comparably for bearing, providing estimates within a few degrees of each other. However, in terms of range, the best performing tracker was MedianFlow.

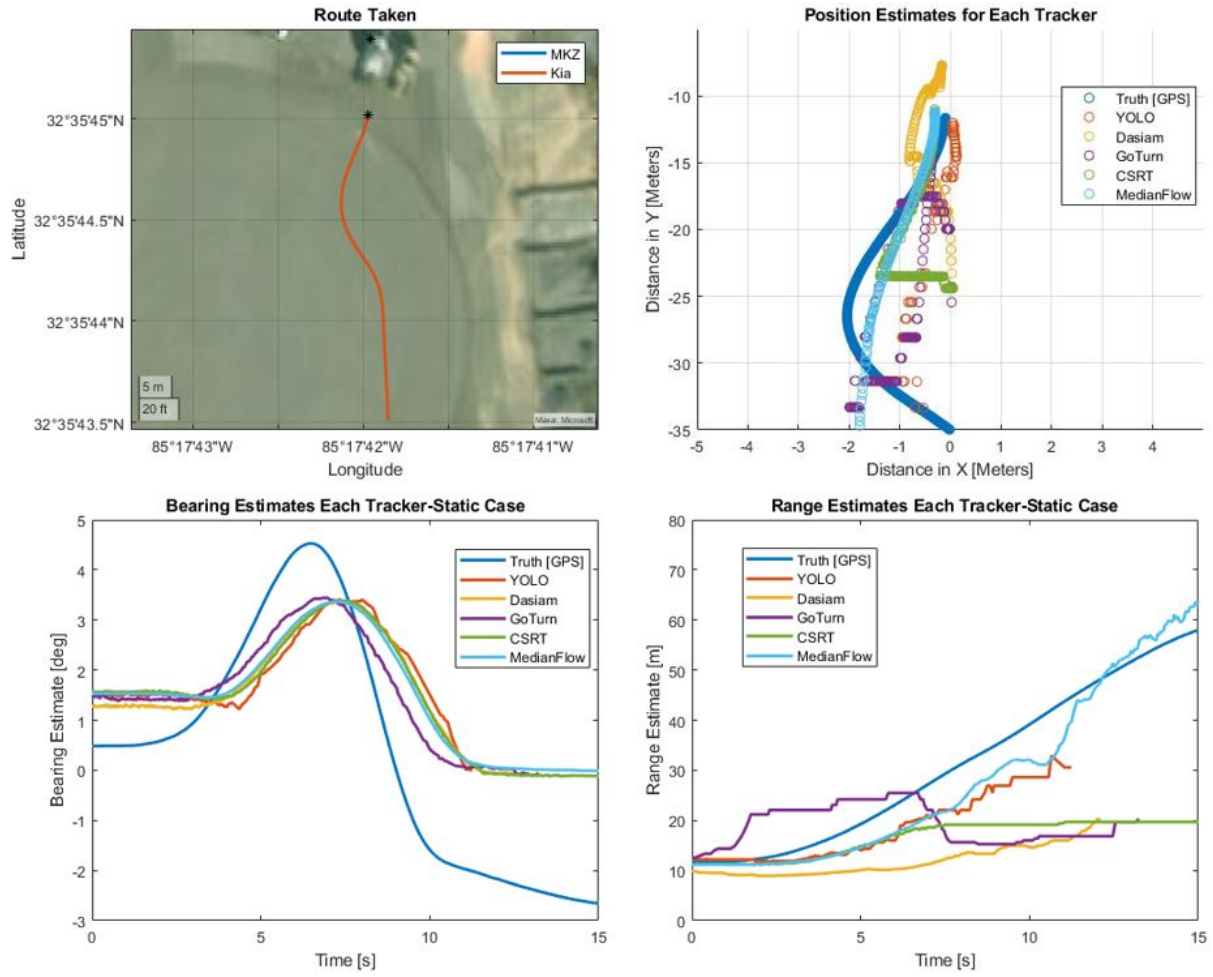

Figure 4.8: Results of Static Tracker Comparison, showing the vehicle path, position estimates, and range and bearing estimates.

The route taken in the dynamic run can be seen in Figure 4.9. Additionally, the range and bearing measurements provided by the trackers can be seen in Figure 4.10. Similar to the static run, in the dynamic case most of the trackers provided comparable bearing measurements, with the exception of GoTurn which performed the worst. In terms of range, CSRT most closely followed the truth measurement, with Median Flow also working fairly well. Moving forward with the development, CSRT was chosen as the best option because of its solid performance in the dynamic case, in addition to the fact that it was seen to be more robust.

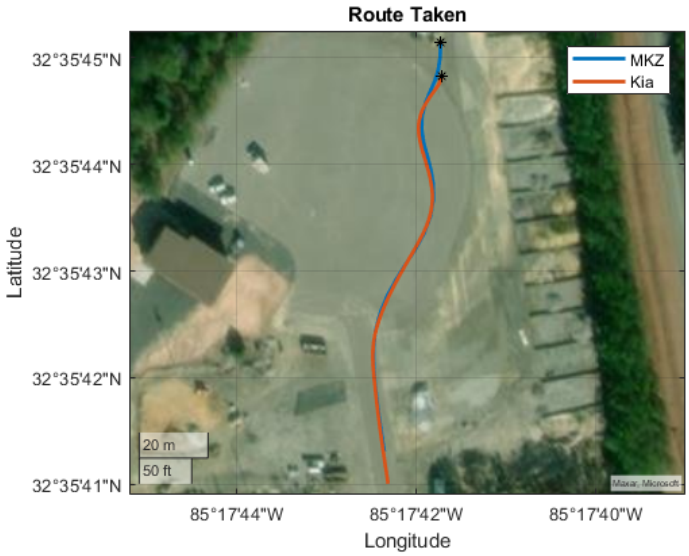

Figure 4.9: Route Taken During Dynamic Tracker Comparison.

4.1.2 Final Algorithm

The final algorithm is depicted in Figure 4.3. The input grayscale image is accepted at 30 FPS, then the CLAHE algorithm is used to normalize the illumination. An attempt to detect the lead vehicle is made using YOLO. If the detection is made, the range and bearing are determined using the models developed in Chapter 3 and the YOLO provided bounding box. If YOLO cannot detect the lead vehicle, the algorithm uses the bounding box from the previous frame to update its position in the new frame using the CSRT tracker, with

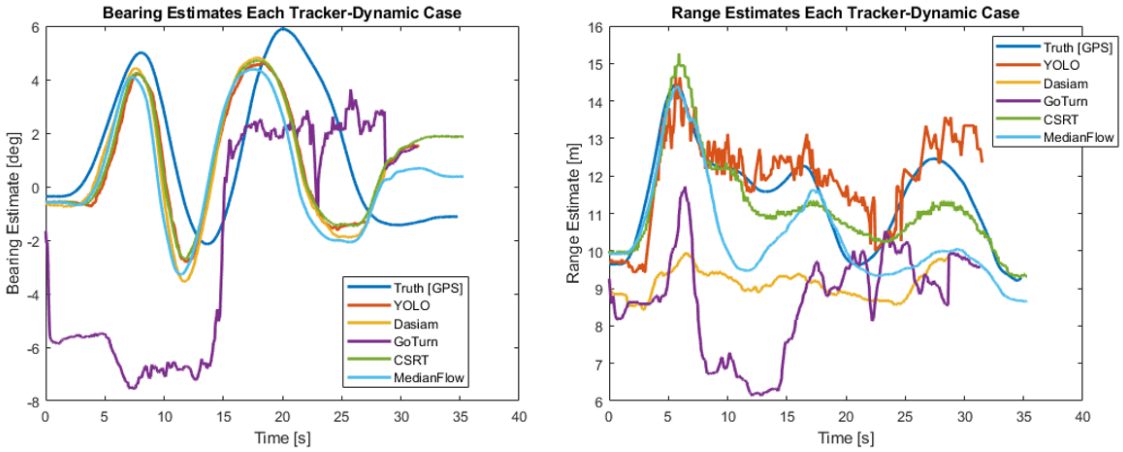

Figure 4.10: Range and Bearing Comparison for Trackers.

this bounding box being used by the range and bearing models instead. The range and bearing are then pushed through a moving average filter prior to providing measurements to the path generation algorithm. Additionally, a visualization tool was created to allow viewing the bounding box, as well as representation of the RPV. An example frame from the visualization tool is provided in Figure 4.11.

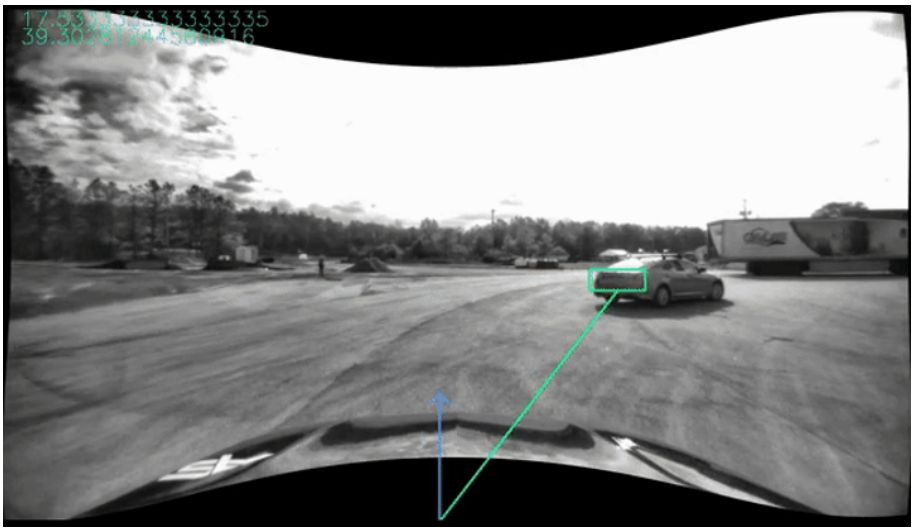

Figure 4.11: Example Output from Monocular RPV Generation Algorithm as Shown by the Visualization Tool.

4.1.3 Stereo Variant

As part of the development, a stereo camera version of the vehicle tracking method for platooning was also considered. A stereo camera varies from a monocular camera because it can provide accurate pixel referenced depth while also providing the traditional monocular image. This means it is possible to detect the lead vehicle in the monocular image, then use the detection to determine range from the depth image. An example of a depth image and its corresponding monocular image can be seen in Figure 4.12. Additionally, the modified stereo camera algorithm can be seen in Figure 4.13. No sub-tracker is used as part of the algorithm. The monocular image provides the bearing measurement, while the depth image provides the range measurement. A comparison between the monocular range estimate and stereo range estimate for a singular pull away run is shown in Figure 4.14. It is shown that the stereo camera range measurement has much less noise and less error than the monocular counterpart.

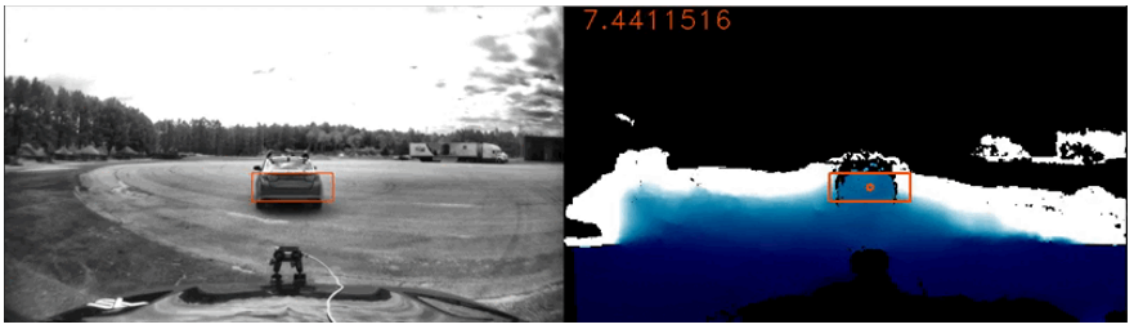

Figure 4.12: Depth Image (left) and Monocular Image (right).

4.2 Results

In order to test the algorithms, a real time test was done with a two vehicle platoon on a skid pad in which the lead vehicle was manually driven in a variety of maneuvers, and the follower vehicle was controlled laterally using camera measurements. Each vehicle featured a Honeywell Etalin II which was used to get a truth RPV for comparison. Three testing

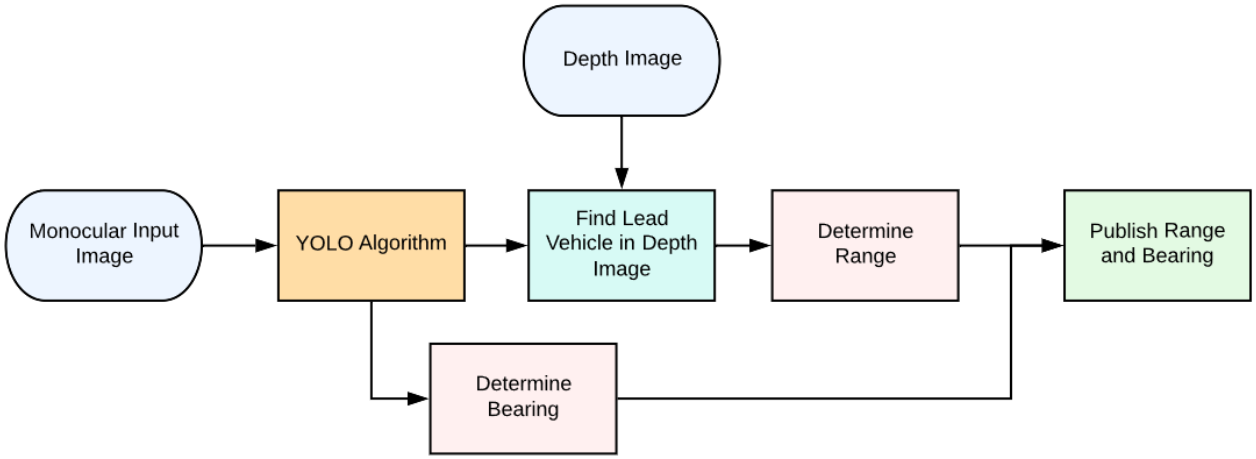

Figure 4.13: Stereo based RPV Generation Algorithm.

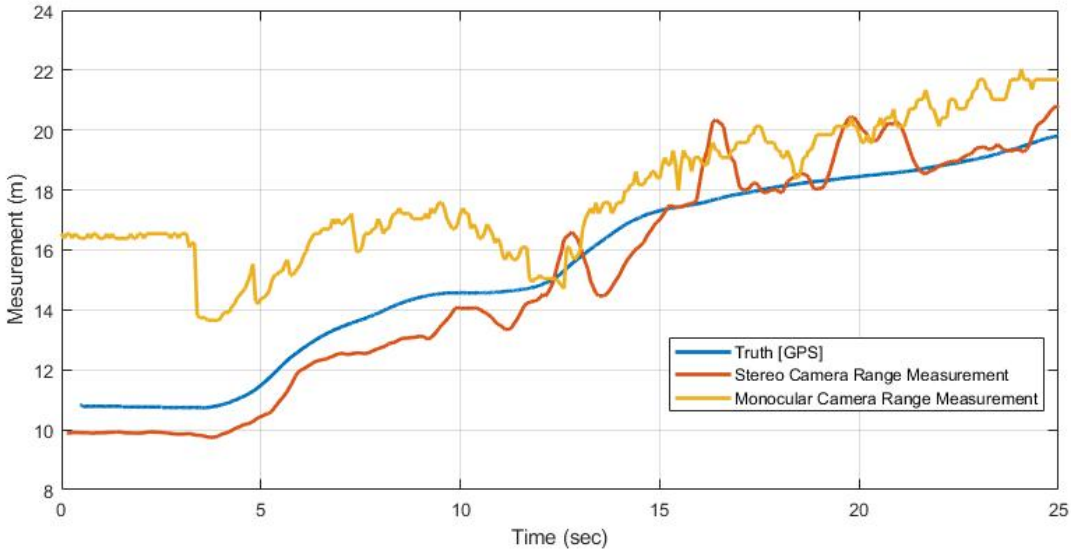

Figure 4.14: Stereo and Monocular Based Range Measurements and GPS Truth

scenarios were considered using different methods discussed in the previous sections. First, a monocular relative path following method was considered. Next, a monocular pure pursuit method was evaluated. Finally, the stereo camera is utilized to determine range in a relative path following scheme.

4.2.1 Monocular Relative Path Following

In the first run type, the objective was to follow the lead vehicle using relative path following, in which the follower vehicle attempts to replicate the exact path made by the lead vehicle using a monocular camera. The paths taken by both the leader (KIA) and follower (MKZ) as well as the X and Y measurements provided by the camera and GPS truth are provided in Figure 4.15. For the GPS truth data, there is a large spike in the X measurement at approximately 27 seconds. It can be seen in the plot the vehicles took during the run, a portion of the path taken by the KIA exhibited an uncharacteristically straight portion. It can therefore be assumed that the error is related to a GPS related issue at this time. This means that at least at this instance, the camera provided the required platooning measurements when GPS could not. At other times in the run, it is shown that the measurements follow the trend of the GPS truth. In Figure 4.16, prior to the GPS issue, the error for X remains below 10 meters. However, in the Y direction, the error is more significant, especially in the turn. While the method worked, and the follower replicated the path of the lead vehicle, there were issues with repeatability due to the lead vehicle leading the image frame, and some issues with measurement error especially in the Y direction.

4.2.2 Monocular Pure Pursuit

In the second run type, a pure pursuit control scheme was utilized in which the input to the controller is heading error, again just using the monocular camera. The path taken by both vehicles can be seen in Figure 4.17 along with the X and Y measurements of the RPV

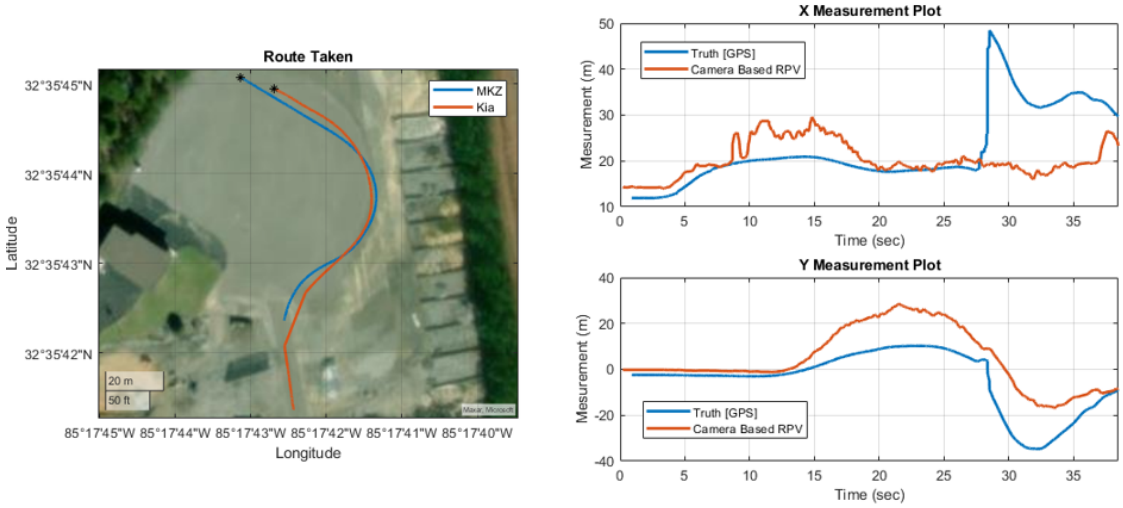

Figure 4.15: Monocular Relative Path Following Results: Path Taken (left) and RPV Generated (right).

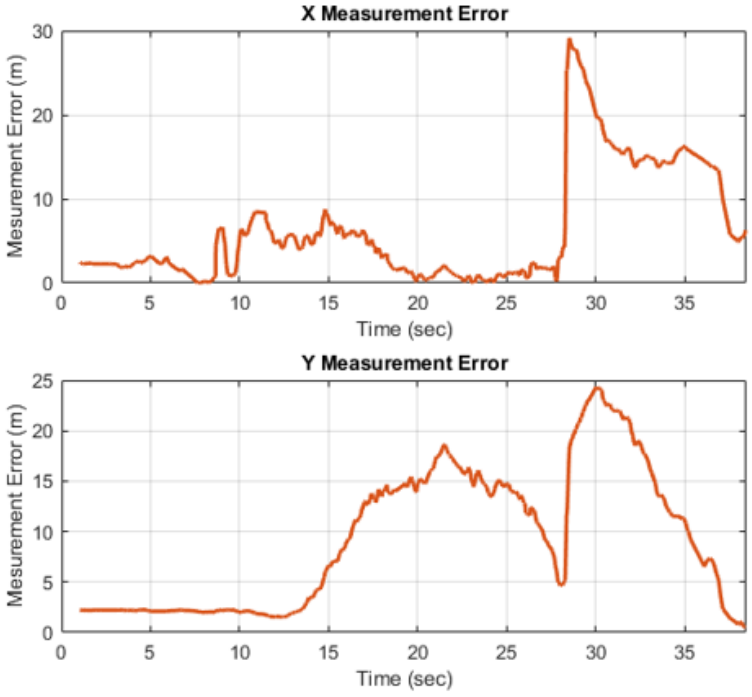

Figure 4.16: RPV Errors from Monocular Relative Path Following Run.

provided by the camera. It can be seen that again the measurement provided by the camera follows the trend of the GPS solution. The highest errors appear to correspond to instances where the leading vehicle gets more than 20 meters away. The pure pursuit method was more stable. This is most likely due to the fact that as the controller was constantly trying to drive heading error to zero, it was also keeping the lead vehicle centered in the image frame. This resulted in a more consistent and stable performance, versus the relative path following which sometimes lost line of sight to the lead vehicle.

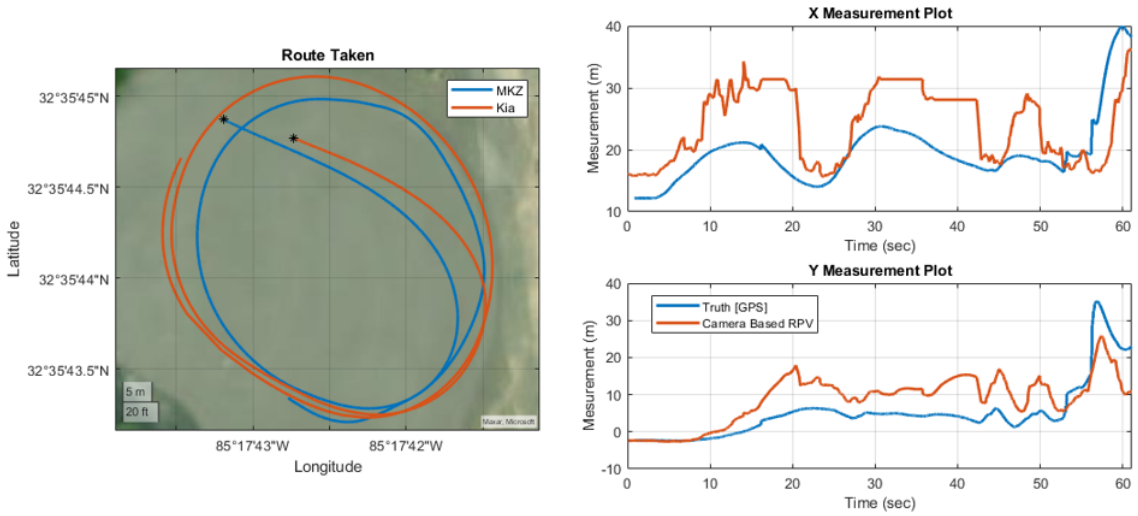

Figure 4.17: Monocular Pure Pursuit Following Results: Path Taken (left) and RPV Generated (right).

4.2.3 Stereo Relative Path Following

In addition to the monocular camera based relative following, the stero based relative path following was also considered. The path taken and X/Y measurement plots can be seen in Figure 4.19. The error is seen to be significantly lower than that of the monocular based method as shown in Figure 4.20. There is a lapse in data for a few seconds likely because the stereo depth image was not dense enough for that specific time window, potentially due to not enough matching features existing to solve the correspondence problem. There is also

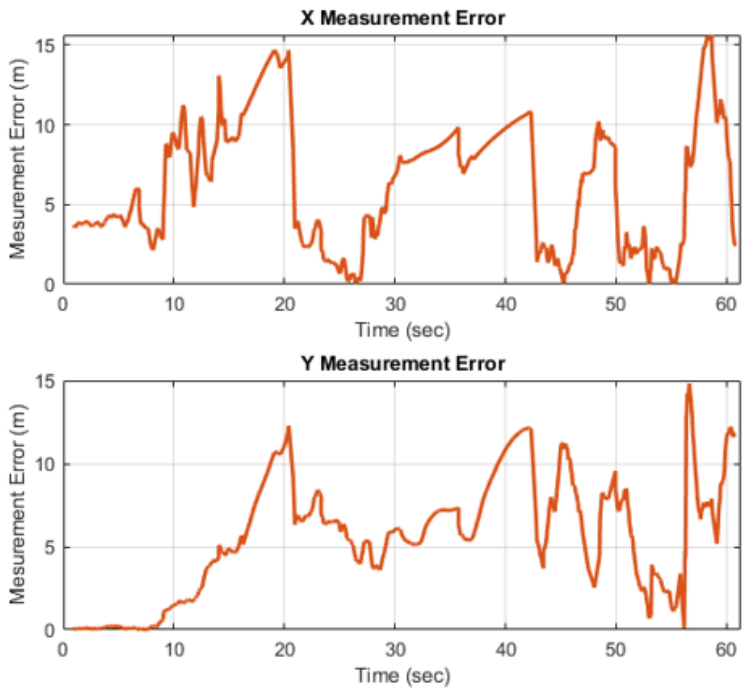

Figure 4.18: Monocular Pure Pursuit Following Run Errors.

a bias seen in the Y measurement. This is likely because the tuning parameters from the monocular camera were used for the data run. It is expected that this could be improved by using data collected to better tune the model to the stereo camera. Similarly, the seemingly constant bias seen in the X measurement might also be removable through tuning. The cause of this bias is likely mainly due to the fact that the GPS RPV is measured antenna to antenna between the two vehicles, and the Camera RPV is measured from the stereo camera to the trunk of the lead vehicle. The measured distance between the antenna and the rear of the trunk of the KIA is approximately 1.75 meters, and the distance between the stereo camera's mounting location and the GPS antenna on the MKZ is approximately 0.9 meters. This bias would cause the measurement of the range in the X direction to underestimate the truth range by approximately 2.65 meters. Even without this tuning, the measurement quality provided by the stereo camera seems to be better than the monocular camera.

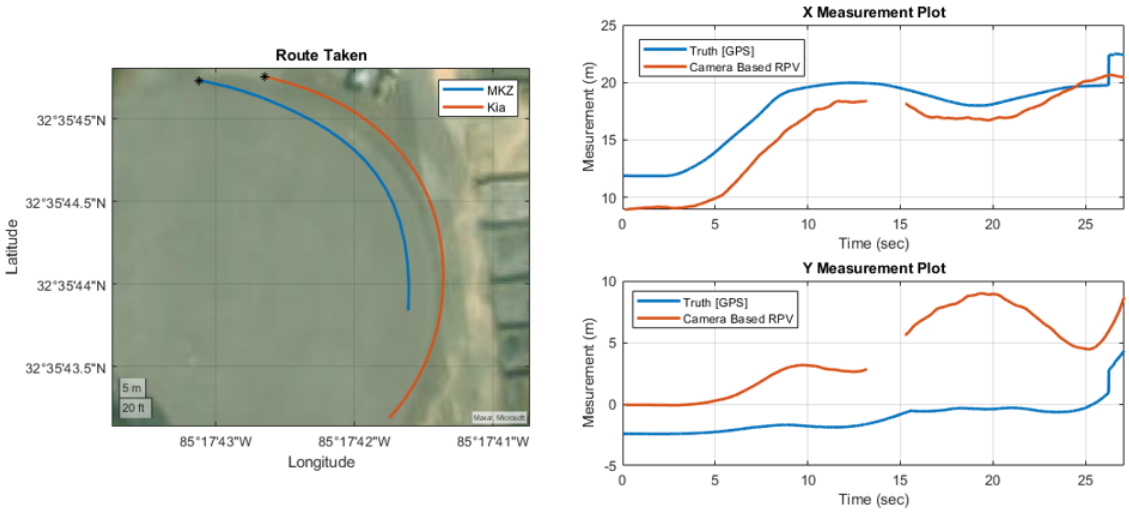

Figure 4.19: Monocular Pure Pursuit Following Results: Path Taken (left) and RPV Generated (right).

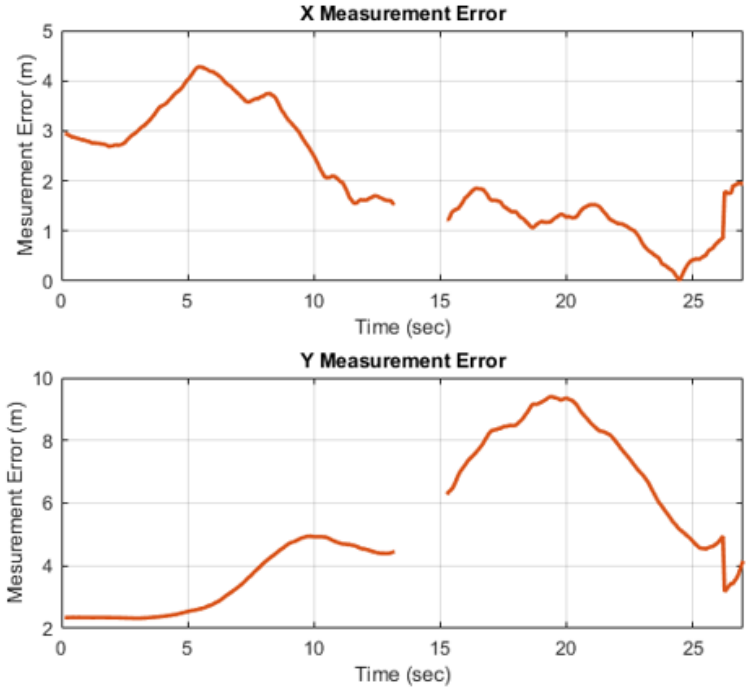

Figure 4.20: Errors Monocular Pure Pursuit Following Run.

4.2.4 Issues Recognized

The largest issue with the methods used in the real time implementation revolve around repeatability and the camera remaining in the image frame. This is likely why the pure pursuit following method worked best from a robustness standpoint. The noisy measurements from the monocular camera also effected performance, even when using a moving average filter. This was also true at longer ranges which saw more error. These noise resulted in a jagged path at times. Another issue discovered during final testing was a slight deviation in the mounting angle of the camera each time. Figure 4.21 shows the difference between the cameras set center parameter shown as the blue arrow, and the center of the hood which is highlighted as a green dashed line. This might appear minuscule, but at longer ranges this slight difference in mount angle can cause a large bias in the bearing measurement specifically. The real-time implementation discussed in this chapter, while promising, has issues that will need to be addressed before the methods can be trusted to perform robustly on roadways. The work presented in this thesis is concluded in the next chapter, with detail provided which will help guide future development.

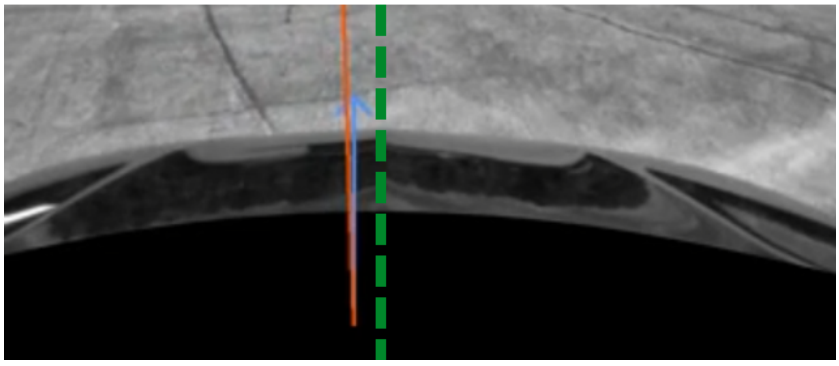

Figure 4.21: Slight Deviation Due to Mounting Differences which Might Contribute To Bearing Measurement Bias.

Chapter 5

Conclusions and Future Work

5.0.1 Conclusion

Due to issues related to GPS outages in platooning situations, this thesis developed and evaluated methods for generating a relative position vector using computer vision. Chapter 3 accessed the ability of varied forms of the YOLO object detection algorithm to detect a heavy duty truck in on road data. These detections were used to generate range and bearing measurements to the lead vehicle, and were compared against DRTK truth. It was found that the YOLO algorithm, when used to generate range measurements, exhibited a mean error of -2.68 meters with a mean standard deviation of 3.83 meters. All YOLO variants and camera selections featured similar noise characteristics. Chapter 4 dealt with the real-time implementations of the methods. An implementation of TinyYOLOv3 was utilized to track a lead consumer sedan by leveraging the pinhole model and bearing model developed in Chapter 3. Following initial testing, it was seen that YOLO sometimes suffered occasional momentary lapses in ability to detect the lead vehicle. To account for this a custom, platooning specific sub-tracker algorithm was created which could use previous YOLO detections to locate the lead vehicle in new image frames. Through the development of the sub-tracking algorithm, the performance of several traditional tracking algorithms were analyzed. The CSRT tracking algorithm performed best on the evaluation data set, and was implemented as part of the sub-tracker algorithm framework. The sub-tracker algorithm was then tested in real-time in several platooning scenarios. While the following vehicle was able to autonomously follow the lead vehicle with lateral control, there were some issues recognized which will need to be addressed in future development. Ideas for future develpment which would resolve these issues are discussed in the following section.

5.0.2 Future Work

The methods developed in this thesis were shown to work in real time. The system was capable of real time following of a lead vehicle without the need for GPS or communications with the lead vehicle. However, there were issues that most be resolved before the methods would be viable for expanded real world use. The biggest issue being the reliability of keeping the lead vehicle in the image frame. The following vehicle drives to a current point on the path without considering the lead vehicles current position or heading. When the following vehicle arrived at the current path point the lead vehicle could be out of frame. This must be solved in order for the methods presented in the thesis to be implemented on road without the use of other complementary sensors. Additionally, the RPV measurements provided were noisy (especially range), even with the moving average filter. This causes problems with the relative path following particularly as it results in a jagged path. Even with these issues, the methods show promise and merit for future work.

Specially Designed Controller / Path Following Scheme

One option to improve the robustness concern of the lead vehicle leaving the image frame, would be to design a custom controller/path following scheme that has a focus on keeping the lead vehicle in frame at all times. This would have to be considered by the controller in addition to the desired path following behavior. If this worked correctly, it would be a major step in solving the robustness issues discussed previously.

Fusion with UWB Radios

Ultra Wide Band [UWB] Radios have been utilized for platooning in prior research [62]. However, they have some noted weaknesses that might be overcome by adding a camera into the solution. First the method requires 4 (or more) UWBs to appropriately determine range and bearing to the lead vehicle, with 2 on each vehicle. The traditional mounting configuration is shown in the top portion of Figure 5.1. Additionally, the UWB Radios have

to be mounted as far apart as possible to increase the observability of the angles between the 4 UWBs. Even with the 4 UWBs there is an ambiguity in which there are two possible solutions at any time, in front of the vehicle or behind the vehicle. It is possible that through the use of the camera, the existing UWB solution might function with fewer UWBs or a tighter mounting configuration might be possible. Additionally, it would also be possible to determine whether the lead vehicle is in front or behind the ego vehicle.

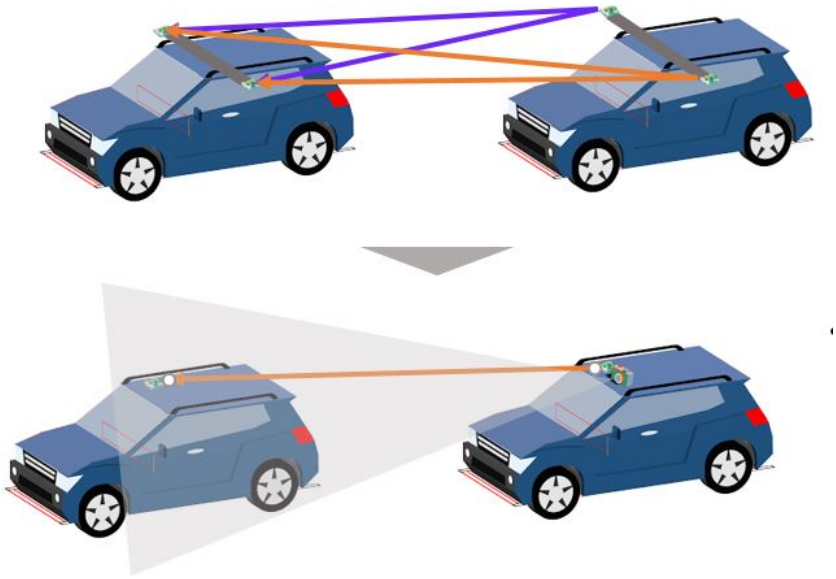

Figure 5.1: Existing UWB Configuration and Potential Reduction in Number of UWBs.

Fusion with IMUs in each Vehicle

Using communications between both vehicles, it would be possible to generate an RPV estimate through mechanizing the IMUs of each vehicle and passing these position changes between vehicles. This RPV would have to be initialized through some other means than the IMUs, and would be prone to drifting overtime. However, the camera based RPV might be able to bound this drift with both the camera and IMUs providing measurement updates to a Kalman filter or similar. It would also be theoretically possible to improve the camera based RPV with a singular IMU on the follower vehicle. A variation of would involve using visual

odometry instead of the traditional inertial measurements used for path following presented in the thesis. This would mean that the only sensor required platooning would be a camera and a velocity measurement.

Fusion with LiDAR or Other Perception Sensors

This thesis has shown that the use of a stereo camera provides performance increases over that of monocular camera when it comes to producing range measurements. This could be extended to LiDAR in which the camera would contextualize point cloud clusters generated by the LiDAR. This would provide a higher fidelity range and bearing measurement. It is also possible that once the lead vehicle has been recognized in the image frame, the LiDAR might be able to track the lead vehicle for a period of time when it is outside of the image frame.

Custom Neural Network Built on Top of YOLO

A dataset can be constructed using camera data and the IMUs in each vehicle to provide the relative pose between the two vehicles. A custom neural network could then be created which accepts the bounded lead vehicle as the input image. A flowchart representing the algorithm is provided in Figure 5.2. The neural network could then trained on the truth relative poses provided by the two IMUs.

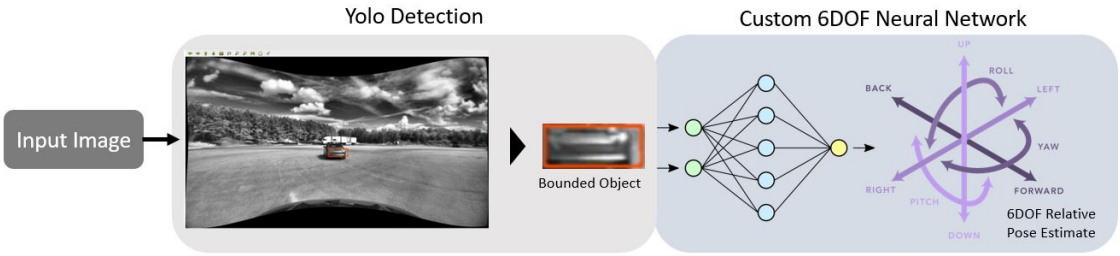

Figure 5.2: Proposed Custom Neural Network Which Provides 6 DOF Relative Pose Information Using Camera and YOLO.

Lane Modeling as a Form of Constraint

A reliable lane model could prove useful for vision based platooning. It could include the follower vehicle using lane boundaries to follow the lead vehicle, and using the camera to keep track of which lane the lead vehicle is in. Another option could be to use the lane estimate to constrain the measurement possibilities in the lateral direction.

Bibliography

[1] Steven E. Shladover. "AHS research at the California PATH program and future AHS research needs". In: *2008 IEEE International Conference on Vehicular Electronics and Safety*. Sept. 2008, pp. 4–5. DOI: 10.1109/ICVES.2008.4640915.

[2] Sadayuki Tsugawa. "Results and issues of an automated truck platoon within the energy ITS project". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. ISSN: 1931-0587. June 2014, pp. 642–647. DOI: 10.1109/IVS.2014.6856400.

[3] Richard Bishop et al. "Results of initial test and evaluation of a Driver-Assistive Truck Platooning prototype". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. ISSN: 1931-0587. June 2014, pp. 208–213. DOI: 10.1109/IVS.2014.6856585.

[4] Sadayuki Tsugawa, Sabina Jeschke, and Steven E. Shladover. "A Review of Truck Platooning Projects for Energy Savings". In: *IEEE Transactions on Intelligent Vehicles* 1.1 (Mar. 2016). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 68–77. ISSN: 2379-8904. DOI: 10.1109/TIV.2016.2577499.

[5] Scott Martin. "Closely Coupled GPS/INS Relative Positioning For Automated Vehicle Convoys". en. Accepted: 2011-04-15T14:18:41Z. thesis. Apr. 2011. URL: https://etd.auburn.edu//handle/10415/2533 (visited on 03/09/2023).

[6] Cetin Mekik and Ozer Can. *An investigation on multipath errors in real time kinematic GPS method*. Aug. 2010.

[7] Ashutosh Saxena, Jamie Schulte, and Andrew Y. Ng. "Depth estimation using monocular and stereo cues". In: *Proceedings of the 20th international joint conference on Artifical intelligence*. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jan. 2007, pp. 2197–2203. (Visited on 03/09/2023).

[8] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". English. In: ISSN: 1063-6919. IEEE Computer Society, June 2016, pp. 779–788. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91. URL: https://www.computer.org/csdl/proceedings-article/cvpr/2016/8851a779/12OmNzFv4de (visited on 07/24/2023).

[9] Jihene Rezgui et al. "Platooning of Autonomous Vehicles with Artificial Intelligence V2I Communications and Navigation Algorithm". In: *2020 Global Information Infrastructure and Networking Symposium (GIIS)*. ISSN: 2150-329X. Oct. 2020, pp. 1–6. DOI: 10.1109/GIIS50753.2020.9248490.

[10] Shankar Kumar. *Platooning Demonstrator with Deep Learning and Computer Vision*. https://expectopatronm.github.io. URL: https://expectopatronm.github.io/posts/projects/deep_learning_projects/platooning/.

[11] Yuto Baba et al. "Development of an Adaptive Cruise Control system using a monocular camera". In: *2011 11th International Conference on Control, Automation and Systems*. ISSN: 2093-7121. Oct. 2011, pp. 1293–1297.

[12] Fidelis Theinert et al. "Object Tracking for 'Car Platooning' Using a Single Area-Scan Camera". In: *2018 19th International Conference on Research and Education in Mechatronics (REM)*. June 2018, pp. 130–135. DOI: 10.1109/REM.2018.8421799.

[13] Samuel Mitchell et al. "Visual distance estimation for pure pursuit based platooning with a monocular camera". In: *2017 American Control Conference (ACC)*. ISSN: 2378-5861. May 2017, pp. 2327–2332. DOI: 10.23919/ACC.2017.7963300.

[14] Simon van der Marel. *Development of a Platform for Stereo Visual Odometry based Platooning — TU Delft Repositories*. College Repository. June 2021. URL: https://repository.tudelft.nl/islandora/object/uuid:1b520436-feed-4a81-8684-ea3d5b1e55e0 (visited on 03/10/2023).

[15] Jakob Dichgans, Jan Kallwies, and Hans-Joachim Wuensche. "Robust Vehicle Tracking with Monocular Vision using Convolutional Neuronal Networks". In: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Sept. 2020, pp. 297–302. DOI: 10.1109/MFI49285.2020.9235213.

[16] Tae-Wook Kim et al. "Camera and Radar-based Perception System for Truck Platooning". In: *2020 20th International Conference on Control, Automation and Systems (ICCAS)*. ISSN: 2642-3901. Oct. 2020, pp. 950–955. DOI: 10.23919/ICCAS50221.2020.9268196.

[17] Wilfried Woeber. "EVALUATION OF DAYLIGHT AND THERMAL INFRA-RED BASED DETECTION FOR PLATOONING VEHICLES". In: *Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, Volume 23, No.1,* vol. 23. Vienna, Austria, 2012.

[18] Carsten Fries and Hans-Joachim Wuensche. "Autonomous convoy driving by night: The vehicle tracking system". In: *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. ISSN: 2325-0534. May 2015, pp. 1–6. DOI: 10.1109/TePRA.2015.7219675.

[19] Carsten Fries, Thorsten Luettel, and Hans-Joachim Wuensche. "Combining model- and template-based vehicle tracking for autonomous convoy driving". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587. June 2013, pp. 1022–1027. DOI: 10.1109/IVS.2013.6629600.

[20] Carsten Fries and Hans-Joachim Wuensche. "Monocular template-based vehicle tracking for autonomous convoy driving". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Sept. 2014, pp. 2727–2732. DOI: 10.1109/IROS.2014.6942935.

[21] Michael Manz et al. "Monocular model-based 3D vehicle tracking for autonomous vehicles in unstructured environment". In: *2011 IEEE International Conference on*

Robotics and Automation. ISSN: 1050-4729. May 2011, pp. 2465–2471. DOI: 10.1109/ICRA.2011.5979581.

[22] John M. Pierre. "Incremental Lifelong Deep Learning for Autonomous Vehicles". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. ISSN: 2153-0017. Nov. 2018, pp. 3949–3954. DOI: 10.1109/ITSC.2018.8569992.

[23] Muhammad Abdul Haseeb. *DisNet: A novel method for distance estimation from monocular camera**.

[24] Sinan Khwandah. "Traffic Signs Recognition and Distance Estimation using a Monocular Camera". In: *The 6th International Conference Actual Problems of System and Software Engineering* (Dec. 2019). Publisher: CEUR Workshop Proceedings (CEUR-WS.org), pp. 407–418.

[25] Qing Xu and R. Sengupta. "Simulation, analysis, and comparison of ACC and CACC in highway merging control". In: *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*. June 2003, pp. 237–242. DOI: 10.1109/IVS.2003.1212915.

[26] Ki-Yeong Park and Sun-Young Hwang. "Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System". en. In: *The Scientific World Journal* 2014 (Jan. 2014). Publisher: Hindawi, e923632. ISSN: 2356-6140. DOI: 10.1155/2014/923632. URL: https://www.hindawi.com/journals/tswj/2014/923632/ (visited on 03/10/2023).

[27] Ahmed Ali et al. "Real-time vehicle distance estimation using single view geometry". In: 2020, pp. 1111–1120. URL: https://openaccess.thecvf.com/content_WACV_2020/html/Ali_Real-time_vehicle_distance_estimation_using_single_view_geometry_WACV_2020_paper.html (visited on 03/10/2023).

[28] Joeseph Redmon. *Darknet: Open Source Neural Networks in C*. 2013. URL: https%20:%20/%20/%20pjreddie%20.%20com%20/%20darknet/.

[29] Vino Mahendran. *Custom object training and detection with YOLOv3, Darknet and OpenCV*. Nov. 2019. URL: https://blog.francium.tech/custom-object-training-and-detection-with-yolov3-darknet-and-opencv-41542f2ff44e.

[30] Jacob Ward et al. *Cooperative Adaptive Cruise Control (CACC) in Controlled and Real-World Environments: Testing and Results*. English. Tech. rep. American Center for Mobility, Aug. 2019. URL: https://www.osti.gov/biblio/1834377 (visited on 03/10/2023).

[31] G.P. Stein, O. Mano, and A. Shashua. "Vision-based ACC with a single camera: bounds on range and range rate accuracy". In: *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*. June 2003, pp. 120–125. DOI: 10.1109/IVS.2003.1212895.

[32] *Camera Obscura*. URL: https://www.nga.gov/press/exh/2866/camera-obscura.html (visited on 04/11/2023).

[33] Jacob Ward. "Methods of Optimal Control for Fuel Efficient Class-8 Vehicle Platoons Over Uneven Terrain". en. In: (Aug. 2022). Accepted: 2022-08-18T20:33:59Z. URL: https://etd.auburn.edu//handle/10415/8431 (visited on 06/28/2023).

[34] William Grant Apperson. "Design and Evaluation of Cooperative Adaptive Cruise Control System for Heavy Freight Vehicles". en. In: (Dec. 2019). Accepted: 2019-12-12T17:18:05Z. URL: https://etd.auburn.edu//handle/10415/7069 (visited on 06/28/2023).

[35] Patrick Smith. "Evaluation of Platooning Efficiency for Heavy Duty Trucks using Cooperative Adaptive Cruise Control". en. In: (May 2020). Accepted: 2020-05-20T17:53:46Z. URL: https://etd.auburn.edu//handle/10415/7241 (visited on 06/28/2023).

[36] Tabea Tietz. *Thomas Wedgwood – possibly the First Photographer — SciHi Blog*. en-US. May 2022. URL: http://scihi.org/thomas-wedgwood-first-photographer/ (visited on 06/28/2023).

[37] *The Daguerreotype Medium — Articles and Essays — Daguerreotypes — Digital Collections — Library of Congress*. eng. web page. URL: https://www.loc.gov/collections/daguerreotypes/articles-and-essays/the-daguerreotype-medium/ (visited on 06/28/2023).

[38] *From the Camera Obscura to the Revolutionary Kodak — George Eastman Museum*. en. URL: https://www.eastman.org/camera-obscura-revolutionary-kodak (visited on 06/28/2023).

[39] *Charge-coupled device*. en. Feb. 2023. URL: https://www.bell-labs.com/about/history/innovation-stories/charge-coupled-device/ (visited on 06/28/2023).

[40] *Fujix DS-X*. URL: https://www.digitalkameramuseum.de/en/cameras/item/fujix-ds-x (visited on 06/28/2023).

[41] *Key differences between CCD and CMOS imaging sensors*. en-US. 2021. URL: https://www.flir.com/support-center/iis/machine-vision/knowledge-base/key-differences-between-ccd-and-cmos-imaging-sensors/ (visited on 06/28/2023).

[42] Chiman Kwan and Jude Larkin. "Demosaicing of Bayer and CFA 2.0 Patterns for Low Lighting Images". en. In: *Electronics* 8.12 (Dec. 2019). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 1444. ISSN: 2079-9292. DOI: 10.3390/electronics8121444. URL: https://www.mdpi.com/2079-9292/8/12/1444 (visited on 06/28/2023).

[43] *Rolling vs Global Shutter - Learn*. en-US. URL: https://www.photometrics.com/learn/advanced-imaging/rolling-vs-global-shutter (visited on 06/28/2023).

[44] Kenji Hata and Silvio Savarese. *CS231A Course Notes 1: Camera Models*. URL: https://web.stanford.edu/class/cs231a/course_notes.html.

[45] *camera_calibration - ROS Wiki*. URL: http://wiki.ros.org/camera_calibration (visited on 06/29/2023).

[46] Gabriel Taubin. *Camera Calibration - ENGN 2502 3D photography*. 2018. URL: http://mesh.brown.edu/3DP-2018/.

[47] Edward Adelson. *Checker Shadow Illusion*. 1995. URL: http://persci.mit.edu/gallery/checkershadow.

[48] *Color spaces in OpenCV (C++/Python) — LearnOpenCV*. en-US. May 2017. URL: https://learnopencv.com/color-spaces-in-opencv-cpp-python/ (visited on 06/29/2023).

[49] *Contrast Limited Adaptive Histogram Equalization - MATLAB & Simulink*. URL: https://www.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html (visited on 07/05/2023).

[50] Arthur COSTE. *Image Processing*. URL: https://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html (visited on 07/05/2023).

[51] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 679–698. ISSN: 1939-3539. DOI: 10.1109/TPAMI.1986.4767851.

[52] *Image Transforms - Hough Transform*. URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (visited on 07/05/2023).

[53] Adrian Kaehler and Gary Bradski. "Bird's-Eye-View Transform Example". In: *Learning OpenCV 3 - Computer Vision in C++ with the OpenCV Library*. 2017, pp. 695–700. ISBN: 978-1-4919-3799-0.

[54] Young-Woo Seo and Ragunathan Raj Rajkumar. "Detection and tracking of boundary of unmarked roads". In: *17th International Conference on Information Fusion (FUSION)*. July 2014, pp. 1–6.

[55] Adrian Kaehler and Gary Bradski. "Chapter 16 - Keypoints and Descriptors". In: *Learning OpenCV 3 - Computer Vision in C++ with the OpenCV Library*. 2017, pp. 493–585. ISBN: 978-1-4919-3799-0.

[56] Avi Singh. *Monocular Visual Odometry using OpenCV*. URL: https://avisingh599.github.io/vision/monocular-vo/ (visited on 07/05/2023).

[57] Fang Zheng et al. "Improved Lane Line Detection Algorithm Based on Hough Transform". en. In: *Pattern Recognition and Image Analysis* 28.2 (Apr. 2018), pp. 254–260. ISSN: 1555-6212. DOI: 10.1134/S1054661818020049. URL: https://doi.org/10.1134/S1054661818020049 (visited on 07/05/2023).

[58] Nushaine Ferdinand. *A Deep Dive into Lane Detection with Hough Transform*. en. May 2020. URL: https://towardsdatascience.com/a-deep-dive-into-lane-detection-with-hough-transform-8f90fdd1322f (visited on 07/05/2023).

[59] Moataz Elmasry. *Computer Vision for Lane Finding*. en. Apr. 2018. URL: https://towardsdatascience.com/computer-vision-for-lane-finding-24ea77f25209 (visited on 07/05/2023).

[60] Ze Wang, Weiqiang Ren, and Qiang Qiu. "LaneNet: Real-Time Lane Detection Networks for Autonomous Driving". In: *CoRR* abs/1807.01726 (2018). arXiv: 1807.01726. URL: http://arxiv.org/abs/1807.01726.

[61] Vino Mahendran. *Custom object training and detection with YOLOv3, Darknet and OpenCV*. en. Nov. 2019. URL: https://blog.francium.tech/custom-object-training-and-detection-with-yolov3-darknet-and-opencv-41542f2ff44e (visited on 07/05/2023).

[62] Ben Jones et al. "Ground-Vehicle Relative Position Estimation with UWB Ranges and a Vehicle Dynamics Model". en. In: *IFAC-PapersOnLine*. 2nd Modeling, Estimation and Control Conference MECC 2022 55.37 (Jan. 2022), pp. 681–687. ISSN: 2405-8963.

DOI: 10.1016/j.ifacol.2022.11.261. URL: https://www.sciencedirect.com/science/article/pii/S2405896322029044 (visited on 07/05/2023).

[63] Ze Wang, Weiqiang Ren, and Qiang Qiu. *LaneNet: Real-Time Lane Detection Networks for Autonomous Driving*. Tech. rep. arXiv:1807.01726. arXiv:1807.01726 [cs] type: article. arXiv, July 2018. URL: http://arxiv.org/abs/1807.01726 (visited on 07/05/2023).

[64] Tyler Flegel, Howard Chen, and David Bevly. "RPV DETERMINATION FOR HEAVY TRUCK PLATOONING APPLICATIONS USING IR AND RGB MONOCULAR CAMERA". In: *AUTONOMY, ARTIFICIAL INTELLIGENCE ROBOTICS (AAIR) TECHNICAL SESSION*. Novi, Michigan, Aug. 2022.

[65] Hamid-Eddine Bouali, Mourad Zghal, and Zohra Lakhdar. "Popularisation of optical phenomena: establishing the first Ibn Al-Haytham workshop on photography". In: (Oct. 2005). DOI: 10.1117/12.2207764.

[66] *The Niépce Heliograph*. URL: https://www.hrc.utexas.edu/niepce-heliograph/ (visited on 07/25/2023).

[67] *History*. URL: https://www.digitalkameramuseum.de/en/history (visited on 07/25/2023).

[68] *2D Imager Modules - Opticon World - Expert in Barcode Scanning*. en-US. URL: https://opticon.com/product-category/scanning-solutions-with-barcode-scanners/oem-fixedmount/2d-cmos-imager-based-modules/ (visited on 07/25/2023).

[69] *CCD-vs-CMOS-image-sensor*. en-US. URL: https://www.1stvision.com/machine-vision-solutions/2019/07/benefits-of-cmos-based-machine-vision-cameras-vs-ccd.html/ccd-vs-cmos-image-sensor/ (visited on 07/25/2023).

[70] *Rolling Shutter vs Global Shutter sCMOS Camera Mode*. en. URL: https://andor.oxinst.com/learning/view/article/rolling-and-global-shutter (visited on 07/25/2023).

[71] *Camera calibration: Explaining camera distortions :: Ori Codes*. URL: https://ori.codes/artificial-intelligence/camera-calibration/camera-distortions/ (visited on 07/25/2023).

[72] *IHS Image Merge*. URL: https://www.usna.edu/Users/oceano/pguth/md_help/html/tbme9hwz.htm (visited on 07/25/2023).

[73] *Hough Transform: The Most Up-to-Date Encyclopedia, News, Review & Research*. en. URL: https://academic-accelerator.com/encyclopedia/hough-transform (visited on 07/25/2023).

[74] MTank. *From Cups to Consciousness (Part 3): Mapping your home with SLAM*. en. Nov. 2020. URL: https://towardsdatascience.com/from-cups-to-consciousness-part-3-mapping-your-home-with-slam-8a9129c2ed58 (visited on 07/26/2023).

[75] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. Tech. rep. arXiv:1804.02767. arXiv:1804.02767 [cs] type: article. arXiv, Apr. 2018. URL: http://arxiv.org/abs/1804.02767 (visited on 07/26/2023).

[76] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Forward-Backward Error: Automatic Detection of Tracking Failures". In: *2010 20th International Conference on Pattern Recognition*. ISSN: 1051-4651. Aug. 2010, pp. 2756–2759. DOI: 10.1109/ICPR.2010.675.

[77] Alan Lukezic et al. "Discriminative Correlation Filter With Channel and Spatial Reliability". In: 2017, pp. 6309–6318. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Lukezic_Discriminative_Correlation_Filter_CVPR_2017_paper.html (visited on 07/28/2023).

[78] David Held, Sebastian Thrun, and Silvio Savarese. "Learning to Track at 100 FPS with Deep Regression Networks". en. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International

Publishing, 2016, pp. 749–765. ISBN: 978-3-319-46448-0. DOI: 10.1007/978-3-319-46448-0_45.

[79] Zheng Zhu et al. "Distractor-aware Siamese Networks for Visual Object Tracking". In: 2018, pp. 101–117. URL: https://openaccess.thecvf.com/content_ECCV_2018/html/Zheng_Zhu_Distractor-aware_Siamese_Networks_ECCV_2018_paper.html (visited on 07/28/2023).

[80] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-Learning-Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (July 2012). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1409–1422. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2011.239.

[81] Robert Brothers. "Path Following and Obstacle Avoidance for Autonomous Ground Vehicles Using Nonlinear Model Predictive Control". en. In: (May 2020). Accepted: 2020-05-18T21:30:47Z. URL: https://etd.auburn.edu//handle/10415/7236 (visited on 07/28/2023).

[82] *Adjust Image Contrast Using Histogram Equalization - MATLAB & Simulink*. URL: https://www.mathworks.com/help/images/histogram-equalization.html (visited on 07/28/2023).

[83] Jean-Yves Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*. 2001.

[84] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". en. In: *International Journal of Computer Vision* 56.3 (Feb. 2004), pp. 221–255. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000011205.11775.fd. URL: https://doi.org/10.1023/B:VISI.0000011205.11775.fd (visited on 07/28/2023).

[85] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. "LIMO: Lidar-Monocular Visual Odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots*

and Systems (IROS). ISSN: 2153-0866. Oct. 2018, pp. 7872–7879. DOI: 10.1109/IROS.2018.8594394.

[86] Andrew Howard. "Real-time stereo visual odometry for autonomous ground vehicles". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Sept. 2008, pp. 3946–3952. DOI: 10.1109/IROS.2008.4651147.

[87] *Blob Detection Using OpenCV (Python, C++) —*. en-US. Feb. 2015. URL: https://learnopencv.com/blob-detection-using-opencv-python-c/ (visited on 07/28/2023).

Appendix A

Tuning Parameters for Cameras and Models

Table A.1: Pinhole Camera Model

Parameter	AXIS P1214-E (Truck)	FLIR Boson (Truck)	iDS UI-3241LE (MKZ)
L	2016.25	1102.76	475
C_y	0	0	1.7

Table A.2: Virtual Horizon Model

Parameter	AXIS P1214-E (Truck)	FLIR Boson (Truck)
F_c	0.0028	0.0040
H_c	2.5	2.5
W_a	2.6	2.6
C_{VH}	300000	150000

Table A.3: Bearing Model

Parameter	AXIS P1214-E (Truck)	FLIR Boson (Truck)	iDS UI-3241LE (MKZ)
K	1050	600	308
C_b	0.0125	0.6	0.125

Appendix B

Overview of Path Generation and Following

B.0.1 Path Generation

A path generation algorithm is necessary for many autonomous vehicle and robotics applications. For the purpose of this thesis, the path generation algorithms serves to provide a set of discrete path points which allows the follower vehicle to replicate the path taken by the lead vehicle. The algorithm for path generation is represented in Figure B.1. The path generation algorithm accepts the range and bearing measurements to the lead vehicle as well as inertial measurements from an IMU or wheel encoder mounted on the follower vehicle. For the results presented in this thesis path generation and path mechanization is performed in the body frame of the follower vehicle but it is also possible to do path generation in other coordinate frames. At startup, the path generation algorithm drops a path point (X, Y position in the body frame) at the initial position of the lead vehicle. The follower vehicle will begin to guide itself to this path point using a lateral controller detailed in Section B.0.2. As the follower is driving to this path point, the path point's relative position to the follower vehicle is updated using the mechanization from the inertial measurements. Meanwhile, the path generation algorithm is also considering the current position of the lead vehicle relative to most recently added point. If the Euclidean distance between the lead vehicle and the closest path point exceeds a threshold, a new path point is dropped. This threshold determines the density of points in the path. When the follower approaches the current target path point and reaches a certain proximity threshold the path point is removed. Points might also be removed if they exist behind the follower vehicle. Additionally, a new path point is selected as the target path point. The result is path points

being added and removed during the operation, and a target path point being decided by the waypoint manager algorithm.

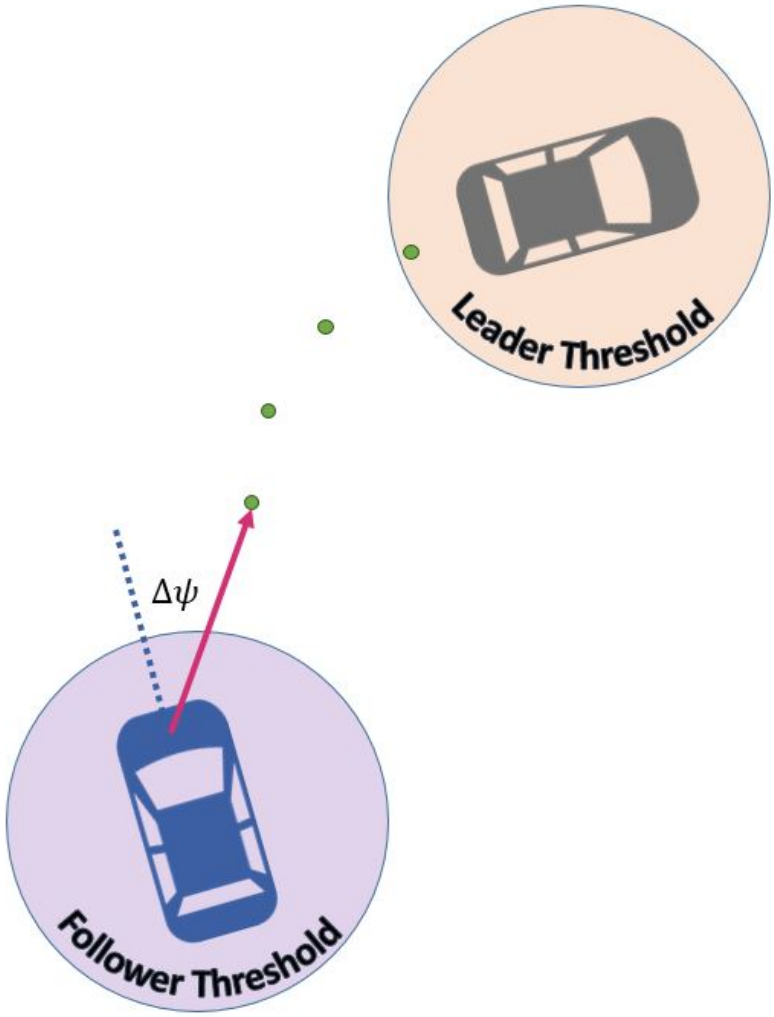

Figure B.1: Path Generation Algorithm Represented Visually.

B.0.2 Waypoint Manager and Lateral Controller

Using the path created by the path generation algorithm, a waypoint manager algorithm decides which waypoint should be the target waypoint. It should be noted that this is not always the closest waypoint to the follower, and is controlled by a "look ahead distance" parameter which determines which waypoint should be used. It should be noted that a large look ahead distance can induce dampening in the system, and can also introduce corner

cutting in turns. The lateral controller itself considers the heading error to the goal path point, and could be done with a feedback controller such as a PID controller.