AUTOCYCLE: DESIGN, CONSTRUCTION, AND VALIDATION OF AN AUTONOMOUS BICYCLE

Michael Allen, Jacob Bartolomei, Jeremy Carter, Cooper Grill,
Mikhail Khrenov, Jack Mirenzi, Joseph O'Leary, Isaac Rose, Evan Ruderman,
Andoni Sanguesa, Logan Swaisgood

Thesis submitted in partial fulfillment of the requirements of the Gemstone Honors Program
University of Maryland
2022

Advisory Committee:
Dr. Romel Gomez, Mentor
Dr. Hosam Fathy
Dr. Dinesh Manocha
Dr. Derek Paley
Dr. Mumu Xu

Abstract

Efficient urban transportation has time and time again proved to be a difficult problem to rectify. One modern solution is the bike-sharing system, where many bicycles are available either at hubs or spread across a city for short-term use. However, usage is limited to those who are located close enough to a bicycle hub that travelling to and from it is time-effective. As for hubless bike-sharing systems, bicycles require redistribution over time to remain conveniently available to many. In this thesis, we propose the concept of a dual-mode bicycle that may either be used by a cyclist manually or operated independently utilizing autonomous locomotion, sensing, and control. Such a bicycle could be implemented into a larger bike-sharing system that autonomously manages balanced redistribution and allows users to summon a bicycle to their location, expanding range of use and encouraging environmentally-friendly transportation solutions in an urban setting. We will explore existing literature that have informed later design choices and data collection methods and propose our own methodology for designing, creating, and testing an autonomous bicycle.

Acknowledgments

We are extremely thankful to our valuable and generous mentor, Dr. Romel Gomez. Also, thank you so much to our librarian, Ms. Nedelina Tchangalova, for her support and time. Thank you to our committee members for advising us throughout the process and participating in our thesis defense. Thank you to Terrapin Works, the IFL, and our donors from the Gemstone Go Fund for providing the necessary resources. Finally, thank you to Gemstone for the opportunity to work on this research.

Table of Contents

Bibliography **146**

List of Tables

List of Figures

List of Symbols and Abbreviations

A	Amperes
ADC	Analog to Digital Converters
AGPS	Assisted Global Positioning System
Ahr	Amp Hours
API	Application Programming Interface
CAD	Computer Aided Design
CAN	Controller Area Network
cm	Centimeters
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPS	Global Positioning System
DC	Direct Current
DGPS	Differential Global Positioning System
DOF	Degrees of Freedom
FOV	Field of View
GB	Gigabytes
GHz	Gigahertz
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
I2C	Inter-Integrated Circuit Bus
ICBM	Inter-Continental Ballistic Missile
IMU	Inertial Measurement Units
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	Light Detection and Ranging
LPV	Linear Parameter Variation
MEMS	Micro Electro Mechanical Systems
MHz	Megahertz
mV	Millivolts
N	Newtons
Nm	Newton-meters
MOI	Moment of Inertia
PDB	Power Distribution Board
PDO	Process Data Object
PETG	Polyethylene Terephthalate (Glycol Modified)

PID	Proportional Integral Derivative
PLA	Polylactic Acid
RADAR	Radio Detection and Ranging
ROS	Robot Operating System
SDK	Software Development Kit
SDO	Service Data Object
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System on a Chip
SONAR	Sound Navigation and Ranging
SPI	Serial Peripheral Interface
STA	Street Tracking Algorithm
TCD	Traffic Control Device
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
V	Volts
YOLO	You Only Look Once
ZSS	Zero Speed Stability System

Chapter 1: Problem Definition and Motivation

Urban transportation projects face many different interconnected variables which significantly increase their complexity [2]. When looking to provide a solution to urban transportation, the criteria that must be considered include but are not limited to speed of transport, range of travel, cost, accessibility, and ease of use. For example, owning a car to get around in the city may provide a great range of travel and be readily accessible at any time, but the cost of ownership is prohibitive to many and the speed of travel may be lesser due to traffic. An option that greatly reduces the cost to travel is public transportation, such as buses or subway systems, however these options are confined to specific destinations and times of departure, among other issues which can reduce the accessibility of travel.

One option that has been steadily growing in popularity within the past few decades is known as a bike-sharing system. First attempted by a community organization in Amsterdam in 1965 and later in Danish cities during the 1990s, the system aimed to provide a large number of bicycles that could be used by anybody in a city to get to their destination, leaving the bicycle for the next user. However, these initial forays into the bike-sharing concept often failed quickly due to user disregard for maintenance and often leaving them in disrepair. In the mid-1990s, Portsmouth University in England implemented the first modern bike-sharing system, where students connected their bike usage to a magnetic card that would hold users accountable for damage

caused to bikes [3]. This concept has been growing in popularity ever since and can now be seen in major cities and campuses across the world.

Bike shares are found in many cities, but most of them struggle with bike redistribution because the distribution of bikes across a city over time does not always match the spread of population using these bikes. The supporting research shows that bike shares reduce car usage and carbon dioxide emissions, and that denser bike racks can combat congestion on other public transit systems [4]. However, a lack of bike distribution often negates these benefits because it can discourage people from using the system at all. One potential solution is hiring operators, who manually redistribute these bikes. There are also currently automatic redistribution systems available for bike-sharing systems, but they are not as consistent and reliable as manual redistribution. This inconsistency is caused by inaccurate demand maps and predictions in the models used for the self-balancing systems. New models have been made based on historical data, but many have not taken factors such as weather into account and still are relatively inaccurate [5]. Bike redistribution operators must use feedback from the balancing systems, and they have found the programs unreliable [6].

One solution that helps bike-shares reduce the distribution challenge among stations is to use a dockless sharing method. This concept has been adopted in many cities already, but community members complain that letting bikes be dumped and scattered along sidewalks and roads lead to safety concerns and unnecessary clutter [7]. There is still a need of convenience for bike shares to be accepted among the public, so if dockless shares are not acceptable, bikes must be available in all docked stations. This means bike distribution is a driving influence for the success of a bike share system [8].

Studies have found that increasing the number of docking stations has only minimally

increased the use of bike shares [9]. As a result, much of the general public still does not take full advantage of them. We hope that by enabling bikes to come directly to users we can greatly increase convenience and so drive use and the corresponding public benefits.

Environmental cleanliness is one of the benefits of the bike-sharing concept over traditional public transport, by providing a form of efficient intra-city transportation that does not emit carbon waste. Another recent technological development that shares this capability is electric transportation, whereby vehicles such as cars are powered using batteries and motors instead of gasoline and engines. Recent adoption of these technologies by companies, such as Tesla or Rivian, have been pursued with the intent of reducing humanity's carbon footprint on the environment [10].

One benefit of electric vehicles is their ability to expand yet another sector of transportation technology: autonomous vehicles. Initial ventures into this idea have been seen most evidently in demos from research institutions and companies such as Tesla, who have adopted semi-autonomous functions into their vehicles, such as autonomous highway travel and lane changing [11].

Upon an initial survey of the existing literature, it seems evident that there is potential worth exploring and adopting electric autonomy into other forms of transportation. This is the guiding purpose behind our research and the subject of this thesis. Specifically, we intend to integrate autonomy with electric bike shares.

Within the past decade, several examples of research and design for an autonomous, electric bicycle have been pursued by research teams at institutions and universities around the world. Most notable are two teams of researchers, one from India and the other from China. A group of undergraduate students from the Indian Institute of Technology Kharagpur (IIT) successfully

developed a prototype of an autonomous electric bike for a design competition from 2015 to 2017, and the paper outlining their design serves as a useful guide in highlighting what subsystems must be considered for our investigations, as discussed further below [12]. Additionally, more extensive research and implementation of obstacle avoidance, guidance using LiDAR techniques, and autonomous bicycle stability was completed by a group of three researchers from China in 2017. They published a paper explaining their ability to avoid obstacles such as objects on the ground and people [13]. Both of these papers prove that the concept of designing an autonomous bicycle is not just speculative but feasible and implementable. The papers also provide a firm foundation for further research into the concept and act as a guide to our research. Although much of the groundwork for our research has been laid by the sources referenced above, much remains to be investigated to continue down the path towards an autonomous bike-sharing system.

Utilizing this prior work as a roadmap, our research team aims design an effective autonomous electric bike-sharing system by using comprehensive telemetric systems to determine experimentally how effective are our eventually chosen methods of autonomy and control. Additionally, further investigation into the possibility of adopting a bike not only for autonomous bike-sharing but also for transportation of goods with minimal design changes could be a new venture not yet fully investigated by these sources.

In the following thesis, each relevant subsystem to a fully autonomous electric bike is investigated by probing a variety of sources, with the goal of developing a comprehensive perspective on the challenges of each individual component's development and integration. These sources will also serve as a backbone of knowledge from which we can begin our research in full. They are organized similarly to the discussion of bicycle design found in the IIT paper discussed above, which provides an extensive listing of all subsystems that we will research [12]. Once a review

of the existing literature has been completed, a detailed report of team Autocycle's design and implementation will demonstrate the progress that has been completed towards the above goals.

Chapter 2: Review of Literature

2.1 Bicycle System Model

To initially simulate the dynamics of our bicycle and later derive a method of controlling its movement, it is necessary to utilize a generalized dynamical bicycle model which involves several key assumptions. Such a model was originally created by [14] in the 19th century, and today, it is still commonplace to use linearized dynamic equations derived from this model as shown in [15].

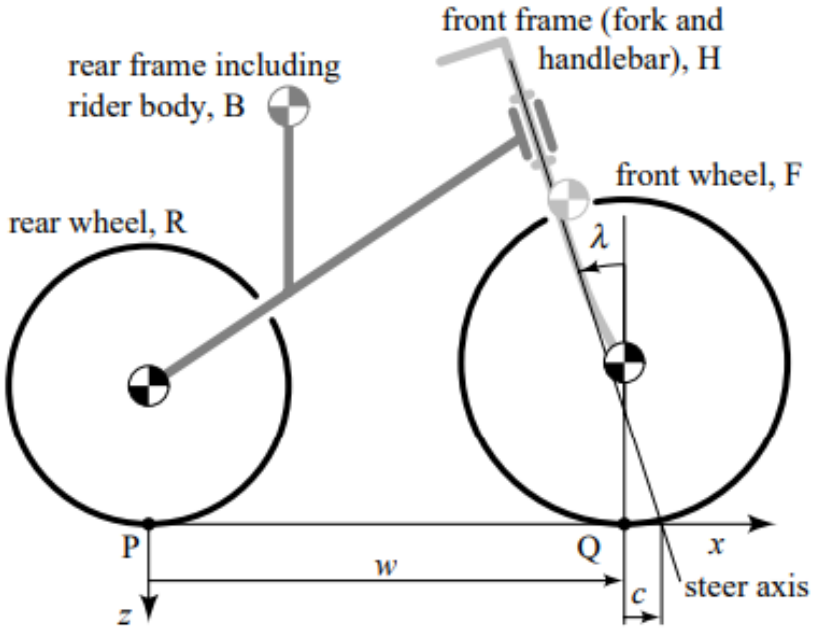


Figure 2.1: Diagram of the modified Whipple model used in [1]. The four rigid bodies are shown - front frame (handlebars), back frame (body), front wheel, and back wheel. Also shown are the following bicycle dimensions: the tilt angle λ, wheel base w, and the trail c.

The modern model presented in [14] considers a bicycle to be a system of three velocity degrees of freedom — forward speed, lean rate, and steering rate — and four rigid bodies seen in 2.1.

The model from [14] presents several generalized assumptions that are important to note when utilizing it for our purposes. The wheels are assumed to be axisymmetric and their contact with the ground is treated as a knife-edge. All four rigid bodies are considered laterally symmetric, and hinges between these bodies are modeled as ideal. Meijaard (et al) derived the entire set of linearized equations of motion for this model that we use in our simulation and experimental control of the bicycle's stability [1]. The emergent behavior of this dynamic model presents some interesting results, including a "self-stable" velocity at which a bicycle may theroetically travel upright without external control as long as it maintains this velocity, simply as a function of its geometric and mass properties.

This model is validated by [16] with some modifications. After measuring the wheelbase, trail, and head angle of the bike (see 2.1); radius, mass, and moment of inertia of both of the wheels; the center of mass, mass, and moment of inertia of both frames (the critical parameters for the modified Whipple model [14]), they were able to experimentally demonstrate behavior in line with the model's predictions for an uncontrolled bicycle, including asymptotic self-stability within a certain range of velocities. It is for this reason that we use of this model and its linearized equations of motion for our design, seen below.

$$M\ddot{q} + vC_1\dot{q} + (K_0 + v^2K_2)q = f \tag{2.1}$$

Kooijman et al define M as a mass matrix, vC as a damping-like matrix, and a stiffness matrix

composed of K0 and K2 [16]. The forces on the right side f are balanced by the degrees of freedom q.

2.2 Bicycle Parameter Identification

The physical parameters of the bicycle that require identification for accurate modeling and control through use of the Whipple model [14] consist of the wheelbase, trail, and head angle of the bike (see 2.1); radius, mass, and moment of inertia of both of the wheels; and the center of mass, mass, and moment of inertia of both frames. Although some of these are basic measurements, the center of mass and moments of inertia in particular require more sophisticated test setups to obtain useful values for these parameters. In their experimental validation of the Whipple model, Kooijman et al utilize a torsional pendulum setup [16]. They also discount the concept of calculating the centers of mass or moments of inertia directly due to the geometrical and material complexity of each part. Each bike part is clamped such that its center of mass is axially aligned with the long, slender metal rod that hangs the bike from above. By twisting the rod to achieve a rotational period, the moment of inertia along the hung axis can be directly calculated using

$$\mathrm{I}_m = (\frac{T}{2\pi})^2(\frac{GI_p}{l}) \tag{2.2}$$

where l is the length of the rod, G is the shear modulus of elasticity, $\mathrm{I_p}$ is the polar moment of inertia of the rod, and T is the rotational period.

Since the entire moment of inertia matrix must be identified, this process is repeated twice more around different axes of the bicycle frame to measure all relevant values. There is a fourth

moment of inertia value, but its relation only to the pitch of the bicycle renders it irrelevant since the bicycle is assumed to have both wheels always in contact with the ground.

2.3 Orientation Sensing

It is critical to the model that the system has accurate values for the bicycle's orientation. The state vector of the model includes the tilt angle, the steering angle, and their respective derivatives. The tilt angle is the rotation of the bicycle along the local x-axis. Inertial Measurement Units (IMU) use Micro Electro Mechanical Systems (MEMS) and Analog to Digital Converters (ADC) to output the orientation of a system. A 6-DOF IMU is a sensor that combines a 3 axis accelerometer MEMS and a 3 axis gyroscope MEMS and can be further combined with a 3 axis magnetometer to get to 9 DOF.

An accelerometer MEMS measures acceleration and utilizes a proof mass surrounded on two sides by springs so that it is able to shift forward and backward in one dimension. The proof mass forms a capacitor with fixed electrodes. Thus there is a change of capacitance relative to its displacement which is relative to the focuses acting on the proof mass. Since the mass of the proof mass is constant the forces are relative to the acceleration of the larger system. Three accelerometer MEMS are combined in orthogonal directions to measure the acceleration of the body in Cartesian coordinates. The global constant downward acceleration due to gravity can be compared to the local acceleration to evaluation the system's orientation.

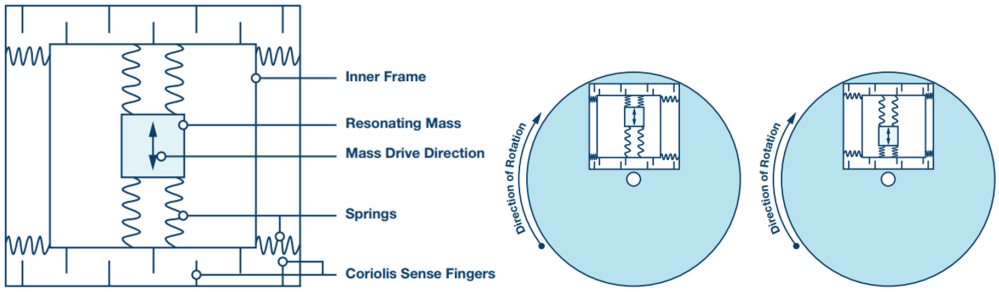


Figure 2.2: Internal MEMS structure of a gyroscope

The gyroscope MEMS measures angular speed and acts very similar to accelerometers but are made to measure the Coriolis effect. A proof mass is set to oscillate along the radial axis of the rotation and force is measured along the tangential axis. When the proof mass is at the peak of its oscillation, furthest from center, it will move in the opposite direction of its tangential velocity due to rotation. If the proof mass is at the trough of its oscillation it will move in the same direction of its tangential velocity due to rotation [17].

2.4 Heading Sensing and Estimation

An autonomous bicycle must be able to tell what direction it is going in the global frame in order to successfully navigate non-trivial distances in the world. Heading can be measured in seven primary ways [18]. A magnetic compass of magnetometer can provide a measurement of the earth's magnetic field, which, with declination values from the World Magnetic Model, can be used to find true north. However, due to the very small magnitude of the Earth's field, these measurement's can easily be skewed by several degrees by ferrous metals, high current power sources, and mineral deposits on land. Gyrocompassing can provide an absolute measurement of true north due to the torque exerted by the rotation of the Earth. While this method has trouble at very high latitudes, the primary issue is the mechanical size and expense inherent to a high

precision gyrocompass.

Heading can also be found by referencing external objects with known position, such as the star trackers used in astro-inertial systems for ICBMs. While such a system can be accurate regardless of latitude and has no moving parts, it requires an unobstructed view of the sky or whatever other reference objects are being used, and being able to detect non-solar stars in daylight requires expensive equipment.

More options are available if the system is equipped with a receiver for a GNSS architecture such as GPS, as this allows the the system to determine its position in global coordinates (latitude and longitude). If an object with a known position is detectable by the vehicle, heading can then simply be calculated by taking the object's position, the vehicle's position, and their relative bearing. Of course, this requires said object to be known, recognizable, and visible (whether optically or over radio frequencies), and thus restricts the functionality of such a system to a given region.

If a vehicle is a large enough, one can simply place two separate GNSS antennas at opposite ends. Heading can then be found by simply taking the vector between the two measured positions. For this method to be accurate, a sufficient separation between the receivers is necessary, and the system must rigid enough for chassis deflections to not introduce erroneous heading offsets.

Finally, if no external references are known and only one GNSS position measurement is available, heading can be reconstructed from vehicle motion. This can be established from the vehicle velocity vector found via finite differences between positions, or the vehicle acceleration vector.

While an absolute reading of heading at rest is appealing, given the size, cost, and mass limitations of the Autocycle prototype, our system used the integrated heading-from-velocity

resource of our GPS receiver, combined with rate gyroscope information to form a Kalman filter.

2.5 Obstacle Sensing and Detection

Vehicles that use local navigation, which is also referred to as reactive navigation, require both detection of obstacles within its local environment and, in the real world, recognition of key objects that produce various different reactions such as stop signs. Object detection and recognition are needed for local navigation, but not for global navigation. Movement within a local system requires perception of surroundings through use of cameras, radar, and/or LiDAR allowing the vehicles to measure out obstacles, calculate their breadth, width, and height in order to calculate how it should move within the local environment to avoid these objects. These sensors can also be used to track the velocity of moving obstacles within the environment and attempt to predict where they will move. This feature allows the sensors on the vehicles to track any moving obstacles that could cause a collision [19].

Through a combination of both global and local navigation, our system must be able to locate itself anywhere on the globe, avoid moving and still obstacles, and retain enough information about its path to make the system's routes safer and more efficient. Past researchers have developed obstacle avoidance algorithms and software that would serve a similar purpose to our local navigation system [20–22]. Ultimately, the approaches to all of these systems can vary quite significantly, with respect to each method of global navigation, each method of local navigation, and each package of obstacle avoidance algorithms having its own set of strengths and weaknesses.

The detection of obstacles on the road is an integral part of an autonomous bicycle in order

to provide the necessary input to the control and stability functions of the bike in order to avoid any obstacles in its path. The challenge of detecting road objects on moving vehicles has been thoroughly studied; however, very little of this work has focused on bicycles. Those that have used a wide variety of detection methods could potentially be implemented on the bike to support its autonomous function.

In order to achieve obstacle detection, the bike will need to detect 2D and 3D objects. In Chen et al's [20] literature, they overview their advancements in an accurate 3D object-identifying software by using monocular systems. Since their research is partially funded by Toyota, they must meet the Karlsruhe Institue of Technology and Toyota Technological Institute (KITTI) Vision Benchmark Suite standards, which are benchmarks for computer vision research set forth by Annieway [20]]. As with all object detection, establishing benchmarks is an important in the development of local navigation. Trying to learn 3D depth from images is a well-known problem in computer vision systems and applications since computers do process visual information differently than humans. Using monocular cameras is a cheaper alternative to the LiDAR software, however, these single-lens cameras have limitations. Some systems use binocular cameras that take multiple pictures enabling depth perception, but single monocular depth perception is inherently harder.

Chen et al's [20] work is primarily centered around creating software that can give a more detailed depth representation than LiDAR software. They propose to detect 3D objects by placing and scoring 3D bounding boxes on images by using semantics, context, size, shape, and location. To achieve this, their software analyzes possible 3D objects by instance-level segmentation, shape, and context. These possibilities are then scored by most promising. In this article, they created software that uses semantic segmentation, shape, instant segmentation, context, and

location to classify objects. Semantic segmentation classifies different objects by counting the number of pixels in the object. Shape is also considered as it analyzes the contour lines. We should be wary of objects that do not create strong contour lines, such as branches in grass or hazards that can blend into the environment. Instant segmentation is segmentation inside and outside the box, and is better for large-scale objects. Context and location are included so it can differentiate between objects, the road, and where the camera is placed.

Once the object is preliminarily scored, it then is further scored by a Convolutional Neural Network (CNN). This creates a faster identifying software. The CNN classifies objects into three different classes: pedestrians, cars, and cyclists. These classes are determined by the KITTI benchmarks. These benchmarks are important features but in the context of an autonomous bicycle, detecting only three classes of objects is not sufficient. Utilizing a schema that detects stop signs, sidewalks, and other objects is necessary for any autonomous vehicle hoping to navigate an urban environment. Chen et al [20] tested their software on KITTI images, which are specified on the KITTI website, and these images can contain up to 15 cars and 30 pedestrians [20]].

To accurately avoid obstacles such as people, sidewalks, and road hazards, the image data must have some type of processing software to distill this data into information that can be acted upon. It is important that the software can work swiftly.

Another method of obstacle detection, and the subject of Alqudah et al's research, is the logging of existing data collected from cellular devices to build a database of known locations and their road hazards [21]. In their research Alqudah et al used a number of sensors already present in cell phones including accelerometers, gyroscopes, magnetometers, and GPS to log sharp changes in acceleration that corresponded to road obstacles. They concluded that this method of keeping track of road obstacles with existing data was effective and accurate to known obstacles on the

road [21]. Implementation of such a system requires a large database and countless hours of surveying. Although Alqudah et al's method seems promising and could be an invaluable tool for future implementations, it would not be a viable option for this project [21].

At Kindai University, Japan, Taniguchi et al. conducted research on the implementation of ultrasonic sensors on bicycles to detect objects and pot holes [22]. Their goal in this research was to allow the bikes to alert riders of obstacles in their path however the method could be used to alert an autonomous bike system as well. Their experimental results showed that the bike, moving at 10 km/h could detect medium-sized objects from 150 cm to 300 cm away, giving an autonomous system, on average, 0.8 seconds to respond and avoid the object. Similar efficacy was found for the sensors ability to detect holes. Unlike Alqudah et al.'s research, this method of object detection is able to function without the need for an existing database and is able to take natural road variance and debris into account [22]. As a result, the method researched by Taniguchi et al seems to be a very promising alternative to Alqudah et al's due to its superior efficacy and low cost. An issue this work does not address is the local path planning aspect as a self-stable bicycle will need a greater warning than .8 seconds to react to an object in its path, avoid it, and remain upright. This solution may have a place as an additive process to increase the efficacy of our object detection, but it lacks the ability to work as the only detection method for our purposes.

Of the many challenges that arise in creating an autonomous bicycle, object classification can be one of the toughest ones to face. While driving to its destination, the bicycle will no doubt encounter pedestrians, stop signs, crosswalks, and other cars. Each of these obstacles requires a different set of instructions for the bike. Consequently, it becomes necessary for the bike to be able to recognize and distinguish these events. Fortunately, the topic of object detection and

classification has been widely studied and has undergone tremendous advancements in speed, accuracy and power efficiency.

There are two main sections of object detection and classification systems, two-stage detectors and one-stage detectors. Two-stage detectors like Faster R-CNN have remarkable accuracy rates, but tend to take longer and are much more intensive on processing. One-stage detectors like YOLO are remarkably fast and resource-efficient, however they lack significantly in accuracy when compared to its two-stage alternative. The lack of accuracy in one stage detectors and speed in two-stage detectors is often attributed to lots of resources being allocated to trying to analyze "background examples", meaning objects or parts of an image which are not what the detector was trained to look for [23]. An autonomous bike would have to react to objects and obstacles in sometimes less than a second. Subsequently, it is clear that speed is a priority in the design of the bicycle, however a lack of accuracy could result in poor fidelity of results.

In order to address the lack of accuracy typical of one stage detectors, recent research has proposed the use of a focal loss system [24]. This system would allow the detector to throw away examples that are easily classifiable as background examples. This research resulted in a new one-stage detector, RetinaNet, that was able to match the speed of traditional one-stage detectors and surpass the accuracy of two-stage detectors. Reaching scores in the high 30s in less than 150 ms on COCO, a standardized benchmark for object detectors, RetinaNet is an optimal candidate for an object detector that could be implemented into an autonomous bicycle [24].

Regardless of the method used to detect objects, the issue of available processing power becomes problematic when trying to adapt these technologies to a moving vehicle, especially with the space constraints present on a bicycle. Minimizing the processing power required to run the detector can physically manifest itself in smaller, lighter, and cheaper parts. To accomplish

this goal, the minimum requirements for detection accuracy must determined. New research by Soviany et al. at the University of Bucharest proposes a method of image difficulty classification. Using a variable difficulty threshold, their algorithm can split images into a 'hard' and 'easy' category. Using this method, the difficulty threshold could be lowered until only the images reflecting the required accuracy are labeled as easy, and the appropriate processing power could be applied to analyze the image [25]. This idea could be utilized to have the correct amount of processing power be allocated and prevent waste of resources.

The sensor that had become the main focus on machine perception for autonomy is LiDAR, Light Detecting And Ranging. LiDAR uses a short pulse of laser then waits to receive back the reflection of the laser using time as a means of measuring the distance [26]. Many of the commercially available automotive grade LiDAR at the time of Autocycle's development were prohibitively expensive or unavailable given our connections. The team was able to identify Livox as a potential LiDAR, though at the time, the LiDAR was not automotive grade. Using a LiDAR, a point cloud can be captured depicting the area around the bicycle. This point cloud is used as an input to a neural networks to detect and classify objects. Using neural network approaches imposes a steep computational cost. The point cloud became the basis of the input for which we worked to develop an algorithm. Raman Maini's paper looks into different methods for edge detection specifically canny edge detection which uses image gradients to find edges [27]. The use of image gradients to find edges or borders of objects would work well on a 2D matrix with distances as the values for cells. Key points could be found doing horizontal and vertical image gradient edge detection to indicate locations of objects but these would just be a group of points. A study by D.T. Lee on an algorithm about finding the convex hull of simple polygons in linear run time presented a potential solution to make sense of key points [28]. Creating convex

polygons for objects will enable better path planning.

2.6 Bicycle Propulsion/Actuation

A common issue that bike-shares face is the fact that the requirement of physical exertion for prolonged travel is a barrier to many potential users. This can be solved through the use of an electric bike. This section will focus on the issue of running an electric bike with sufficient mechanical and electrical power to transport a person with minimal effort on their part. Further, if this is accomplished, as has been done in the past, it would be possible then to apply an understanding of autonomous systems to construct a functional autonomous bicycle. This portion will focus primarily on the selection of a drive motor and mounting system and method by which the motor's power will be delivered to the axle of the bike.

In order to create an electric bike, the mechanical aspect of powering the wheels must be considered. In order to accomplish this, we will use an electric motor. In a paper published by the Alternative Technology Association, the author discusses the use of an electric motor claiming that a hub-mounted motor is lighter and has lower power loss than other types, and brushless motors would be more efficient for this application [29]. Considering these factors, the author decided to use a 12 volt lawn mower motor, sourced from Black and Decker, mounted above the back wheel of the bike and connected mechanically to the rear axle through an angle grinder that he had modified to act as a right angle gear, and through a bicycle chain connected to the angle grinder. This allowed the motor to be geared down such that it provided only 375 rpm to an unloaded wheel rather than the motor's speed 3600 rpm. At that speed, the bike would be unrideable, as the wheel would over spin and lose traction on most types of terrain.

Other sources reviewed have considered other methods with more robust modeling that demonstrate the viability of a centrally mounted motor. According to Abagnale et al., it is possible to use a motor mounted on the lower part of the central frame connected to the pedal axle via either a planetary gear or bevel gear system in a pedelec (pedal electric cycle) bicycle [30]. This literature is primarily concerned with the adaptation of the authors' previous work in developing a pedelec system that is capable of delivering a torque to an actual bike system on an incline or other difficult terrain. The paper's conclusion indicates that the control system from the authors' previous work was successful in allowing for a pedelec bike design that is capable of traversing difficult terrain. However, a pedelec design is not the only method by which an electric bike can be controlled. A handlebar control (like a motorcycle) may be less intuitive for the average user, but may also end up being easier to implement, as will be discussed in the following section.

Meireles et al. noted that for an e-bike sharing system, a centrally mounted motor is more difficult to maintain, however, it is also much less likely to be vandalized or stolen, particularly if it is enclosed within body elements of the bike. In terms of general design, this easily protected motor without the need for long belts or chains to connect motor shaft to the axle of the wheel seems to be more viable, particularly in the area of bike-sharing systems [31]. If a rear-wheel- or front-wheel-mounted motor is used, the maintenance cost will decrease, as the number of moving parts required to access the motor is fewer. The idea of mounting on the rear mudguard as described by Calais would require much more difficult engineering to implement safely and would be even more vulnerable to vandalism and theft than the centrally mounted motor [29].

Several major hindrances exist in selecting a motor and mounting location for an e-bike, particularly one that is intended to be run autonomously include space and power requirements that must be satisfied. In order to have a more powerful motor, more space on the bike must

be allocated to the motor. This motor also must somehow be mounted in a way to minimize interference with a user. If a large motor is used, it takes space that could be used for battery packs, reducing the overall battery capacity of the bike, which may be problematic if the bike is ever left in a position where it cannot return to a docking station for recharging. Thus, the selection of a centrally mounted motor is more constrained than an axial mounting, where the motor is simply connected directly to the axle of either the front or back wheel of the bike.

One of the earliest and most prevalent proposals to accomplish stability is by means of some sort of additional mechanisms, such as an actuated counterweight. The idea is that the weight, by swinging, is able to shift the center of mass in opposition to undesired motion and thus maintain balance, much as a human rider leans to stay stable. This concept has been proposed by several teams and has had its efficacy simulated in several variations, including an upwards facing pendulum, a flywheel, and retractable training wheels [32]. Alvarez et al. were able to demonstrate a PID (Proportional Integral Derivative) based control algorithm for manipulating an inverted pendulum to achieve stability both in motion and at a stop using a simulation, demonstrating the potential of the idea and perhaps warranting an empirical test. While promising, these ideas require the costly addition of an additional mechanical system.

For this reason, especially over the past decade, much research has been done into control schemes which behave much like a human rider: by selectively applying torque to the handlebars. Control algorithms that use this method take the bicycle models previously described and make the torque on the handlebars a controlled quantity rather than a free-floating one. Constraints are applied onto the now actuated system with a goal of achieving stability about some roll angle (usually 0). Guo et al discuss and demonstrate a solution along these lines using Lagrangian methods [33]. Cerone et al maintain roll stability with a linear parameter variation (LPV) ap-

proach [34]. Tanaka et al propose two distinct systems for maintaining both posture control (roll stability) and steering, using steering torque and lateral velocity [35]. Hashemnia et al were able to accomplish roll stability and partial heading stability for a large range of velocities using a genetic algorithm trained fuzzy logic approach with the Lyapunov criterion, demonstrating superior results to Cerone et al when it came to minimizing overshoot and settling time [36]. Simultaneously, Gundes and Nanjangud offer a low order controller for roll stability [37]. Yuan and Sun are able to use a Gaussian pseudospectral method and Yuan and Chen et al are able to use a PSO algorithm to globally minimize errors in both roll and yaw [20]; [38]. The volume of research into this variation of the problem is extremely promising for an autonomous, mechanically simple bicycle, however much remains to be done: in almost all of these cases a range of assumptions about the surface and velocity of the bicycle are made, and the proposed methods are demonstrated in simulation, and not with empirical follow-up.

Methods of steering to keep the bicycle upright work well at stable velocities, but it may not be viable near zero velocity. There are a few promising solutions for these transitional periods. Keo et al were able to build a bike that was stable at zero velocity with the use of a balancer, a large upside-down pendulum, in addition to controlled steering [39]. Other ways to stabilize the bicycle at low speeds are to use arms or drone-like air propellers to keep the bike upright. Both of these would widen the profile of the bike which would make it difficult or impossible to utilize the steering stabilizing technique at higher speeds as well as to switch over to manual control [32]. It is possible to have the arms/propellers retract back into the body of the bike, and many patents have been published with similar ideas. One such patent, involves a sprung set of training wheels that would still allow some steering control [40]. Other patents focus more on the retracting part of the wheels whether ascending upward, [41] or swinging back [42]. A

combination of two or three of these patents could create a bicycle that stayed stable by having two modes: one at operational speed, and the other at zero velocity.

2.7 State Space Control of a Bicycle

Once basic propulsion has been achieved and a mathematical model has been established (see Sec. 2.1, the main control problem that must be addressed for an autonomous bicycle is that of stability. If the bicycle cannot remain upright on its own any navigation capabilities will be for nothing. In this section, we will seek to introduce the current state of work in the field and the variety of proposals to accomplish this task.

One of the earliest and most prevalent proposals to accomplish stability is by the addition of a second actuator, such as an actuated counterweight or reaction wheel. The idea is to exert a torque directly on the roll axis of the bicycle, much as a human rider leans to stay stable. This concept has been proposed by several teams and has had its efficacy simulated in several variations [32], including an upwards facing pendulum, a flywheel, and retractable training wheels. Meanwhile, [43] demonstrate empirically the dynamic effects of a control moment gyroscope on a general unstable system. This implies the potential for such a system's use on a bicycle for stabilization. While promising, these ideas require the costly addition of a mechanical system.

Other concepts for stability take advantage of the special mechanical properties of a bicycle. For one, a bicycle is self-stable in a certain velocity range without any control whatsoever [15, 32, 36, 44]. This can be seen from considering the linearization of the Whipple model dynamics for constant velocity about the zero roll/zero steering equilibria. Solving for the real parts of the system's eigenvalues then shows asymptotic stability in a certain velocity range [1, 16, 36, 44].

Both Kooijman et al and Astrom et al were able to demonstrate these effects. While Astrom et al emphasized the importance of caster and gyroscopic effects in the self-stable regions, in a different work Kooijman demonstrated empirically by adding a counter-gyroscope and negating caster effects that a bicycle could remain self-stable regardless [15]. Unfortunately, while self-stability is a useful phenomenon, it can not be relied upon outside of a very narrow range of velocities, and so can not be the basis of any truly autonomous bicycle.

For this reason, especially over the past decade, much research has been done into control schemes which behave much like a human rider: by selectively applying torque to the handlebars. Most such control algorithms take the bicycle models previously described and make the torque on the handlebars a controlled quantity rather than a free-floating one. Constraints are applied onto the now actuated system with a goal of achieving stability about some roll angle (usually 0). Guo et al discuss and demonstrate a solution along these lines using Lagrangian methods [33]. Cerone et al maintain roll stability with a linear parameter variation (LPV) approach [34]. Tanaka et al propose two distinct systems for maintaining both posture control (roll stability) and steering, using steering torque and lateral velocity [45]. Hashemnia et al were able to accomplish roll stability and partial heading stability for a large range of velocities using a genetic algorithm trained fuzzy logic approach with the Lyapunov criterion, demonstrating superior results to Cerone et al when it came to minimizing overshoot and settling time [36]. Simultaneously, Gundes et Nanjangud offer a low order controller for roll stability [37]. Meanwhile, Yuan and Sun are able to use a Gaussian pseudospectral method and Yuan and Chen et al are able to use a PSO algorithm to globally minimize errors in both roll and yaw [20, 38]. The volume of research into this variation of the problem is extremely promising for an autonomous, mechanically simple bicycle. However, much remains to be done: in almost all of these cases a range of

assumptions about the surface and velocity of the bicycle are made, and the proposed methods are demonstrated in simulation, but not with empirical follow-up.

These methods of steering to keep the bicycle upright work well at speed but may not be viable near zero velocity. There are a few promising solutions for these transitional periods. Keo et al were able to build a bike that was stable at zero velocity with the use of a balancer, a large upside-down pendulum, in addition to controlled steering [39]. Other ways to stabilize the bicycle at low speeds are to use arms or drone-like air propellers to keep the bike upright. Both of these would widen the profile of the bike which would make it difficult or impossible to utilize the steering stabilizing technique at higher speeds as well as to switch over to manual control [32]. It is possible to have the arms/propellers retract back into the body of the bike. Many patents have been published with similar ideas one patent, [40], involves a sprung set of training wheels that would still allow some steering control. Other patents focus more on the retracting part of the wheels whether ascending upward, [41] or swinging back [42]. A combination of two or three of these patents could create a bicycle that stayed stable by having two modes: one at operational speed, and the other at zero velocity.

While the extensive theoretical work that has been done in terms of stability is promising, much remains to be done for this project. There are recordings and demonstrations of implemented stably controlled bicycles, but it is difficult to relate them to specific literature, and most literature demonstrates its validity by simulation. Further, very little to no research has been done into the stability of a bicycle system on a non-level surface, and stability while following a path across an uneven surface is virtually unexplored territory, as is the transition from stability at zero-velocity, to stability at speed. While this project is not able to address all of these open questions, we believe our practical control implementation can shed new light on realities

involved.

2.8 Path Planning

In addition to stability, a central goal for the bicycle is that it should have the capacity to navigate from some initial position to another. To do so, the bicycle must be able to avoid obstacles detected in its vicinity while attempting to reach the desired destination. This problem can be broken down into three parts:

1. Planning short path immediately around obstacles.

2. Smoothing and augmenting the short path using interpolation.

3. Planning a long path from the start point to the desired end point.

Chu et. al. [46] addresses short distance path planning around obstacles using a graph traversal method. In this paper, Chu et. al. propose two different kinds of graphs to traverse. The first is a square grid, where a node is connected to a another node in each cardinal direction by a straight line. In the second graph, each node is connected to three other nodes directly in front of the start node, one straight, and two slightly off to each side of the node. The nodes to the side are connected by a curve. The initial node begins at the position of the vehicle and initially grows in the direction the vehicle is facing. This second graph is meant to take the kinematics of a moving vehicle into account. Obstacles are overlaid with the graph and any nodes that lay over the objects had their connected edges removed. Once invalid edges are pruned, the researchers employ an A^* graph traversal approach to generate path to some predetermined end-point. This path is not guaranteed to be the shortest one.

An alternative, more sophisticated system is proposed by Lim et. al. [47]. Unlike the algorithms proposed by Chu et al. [46], this system takes into Traffic Control Devices (TCDs) into account. These TCDs might include speed bumps, lane markers, street lights, etc. The first step in the process is to generate a set of candidate maneuvers that could bring the vehicle from a start point to some desired end point in sight. These paths are then created taking obstacles and TCDs into account. Similar to the Chu et. al. [46] algorithms, Lim et. al. [47] utilizes a modified A^* approach. The main difference is that a curvilinear coordinate system is used rather than a Cartesian one. In this coordinate system, possible points are added as nodes for the graph traversal and those interesting with static objects are pruned.

One crucial area also addressed by Lim et. al. [47] is the need for disparate local route planning and global route planning. If the desired end position for an autonomous vehicle is miles away, it would be an intractable task to immediately generate a viable path accounting for obstacles, TCDs, and road curvature. To account for this the local route planning described above must query a global route planning system for intermediate desired points. This global route planning system would require both GPS and knowledge on the road or terrain layout.

In application, integrating Global Positioning System (GPS) functionality would allow the bicycle to drive from a hub to the user and back after their trip is completed. It must follow traffic laws, choose optimal pathing, and arrive at its destination within a significantly low margin of error. This challenge poses many unique problems and hurdles that are not accounted for in the local navigation systems that address avoidance of objects and pathing on a smaller scale. To integrate global navigation to a level of sophistication that an autonomous bicycle would require, the navigation system would have to account for challenges with GPS accuracy, GPS reception, and route selection, among other technical obstacles.

Using GPS, despite its deficiencies in accuracy and reception, is going to be impossible to avoid when trying to integrate global navigation capabilities. GPS information, unlike local sensors, are easily interfaced with existing mapping technologies that allow for localizing the bike onto roads and enable optimization of route planning. Since GPS is a requisite technology for an autonomous vehicles, the question becomes what kind of GPS should be integrated into Autocycle [48]. There are three major options for GPS available on the market, standard GPS, Differential GPS (DGPS), and Assisted GPS (AGPS). All three have different common use cases and different levels of accuracy, but also vary greatly in availability. Basic GPS utilizes triangulation information from various satellites orbiting the Earth. GPS has the widest area of availability, pretty much anywhere in the world, however GPS also has the lowest accuracy at around a 60-300 foot radius [48]. AGPS, utilizes ground relay stations to correct for errors in satellite data. This increases accuracy to a 50-150 foot radius and even less if there is a clear line of sight to the relay station [48]. Finally, DGPS which works by contacting as many satellites, roving and stationary relay stations as possible, has by far the best accuracy of about a 3 foot radius, but this technology comes at a significant monetary cost and is not ubiquitous across the US [48].

When using a GPS system, GPS error and data drift during GPS outages must be accounted for. Consequently, addressing and adjusting for GPS error is a highly studied topic that has resulted in numerous solutions with varying degrees of success. The common factor between most literature regarding GPS error correction is the use of onboard motion sensors, particularly gyrometers to measure rotational velocity and acceleration and accelerometers to measure linear acceleration. The use of onboard sensors can greatly increase the accuracy of GPS systems. Specifically, the sensor output can be fed into a Kalman filter which is able to dynamically update

a model of the vehicle's movement, including its speed and direction. The use of just these two onboard sensors and a Kalman filter, location accuracy can increase by as much as 20% [49].

In addition to these systems, a camera tied to an object classification system can be used to greatly improve GPS data, particularly when error is due to data drift. The way this system works is by using an existing map database and providing regular lateral and horizontal localization corrections. The object classification, in this case, is trained to detect and classify lanes and crosswalks. The detection of these objects along with their distance from the autonomous vehicle can be used and compared with existing map data and the predicted location of the vehicle in order to provide lateral (crosswalks) and horizontal (lane) corrections to that predicted location [50]. Along with the use of the other on-board sensors, this setup provides a particularly precise location for the vehicle on a map model. Additionally, what makes this prospect particularly promising for the autonomous bicycle, is that cameras and object classification algorithms are already planned additions for use in local navigation. This would allow the team to re-purpose existing technology on the bicycle for use in this portion of the project.

The final popular method of GPS error correction is through the use of a street tracking algorithm (STA). This system uses lane information similar to that used in horizontal and vertical corrections, however, it is used in a slightly different method. In typical STAs, lane geometry is compared to known map geometry to match the vehicle's location on a road in a given map model [51]. This appears similar to the original use of lane data but differs in a few significant ways. The first uses lane data corrects GPS in the direction perpendicular to the road direction by essentially using information on which lane it is in. The lane data can also be used to localize the vehicle by cross-checking the road geometry to known roads.

Some combination of the error correction methods mentioned are going to be a necessary

part of the global navigation for the autonomous bicycle. The use of these methods would be one step in enabling true autonomous navigation for the bicycle. Once localization data is accurate, the only main obstacle facing an autonomous vehicle from a purely navigational standpoint is route planning which requires very different assumptions and considerations when there is a bicycle involved and not a car.

Chapter 3: System Design & Implementation

The primary goals of the Autocycle system, as previously stated, were to move the bicycle autonomously from one position to another. A wide variety of systems and subsystems were necessary to accomplish this task, with their own design considerations. In this section we present the design choices made and the principal aspects of our implementation of the Autocycle prototype.

For the reader's reference, Fig. 3.1 presents a simplified overall block diagram of the Autocycle system's operation and the interaction of the various sub-systems, while Fig. 3.2 shows the location of the various sub-systems on the assembled prototype.

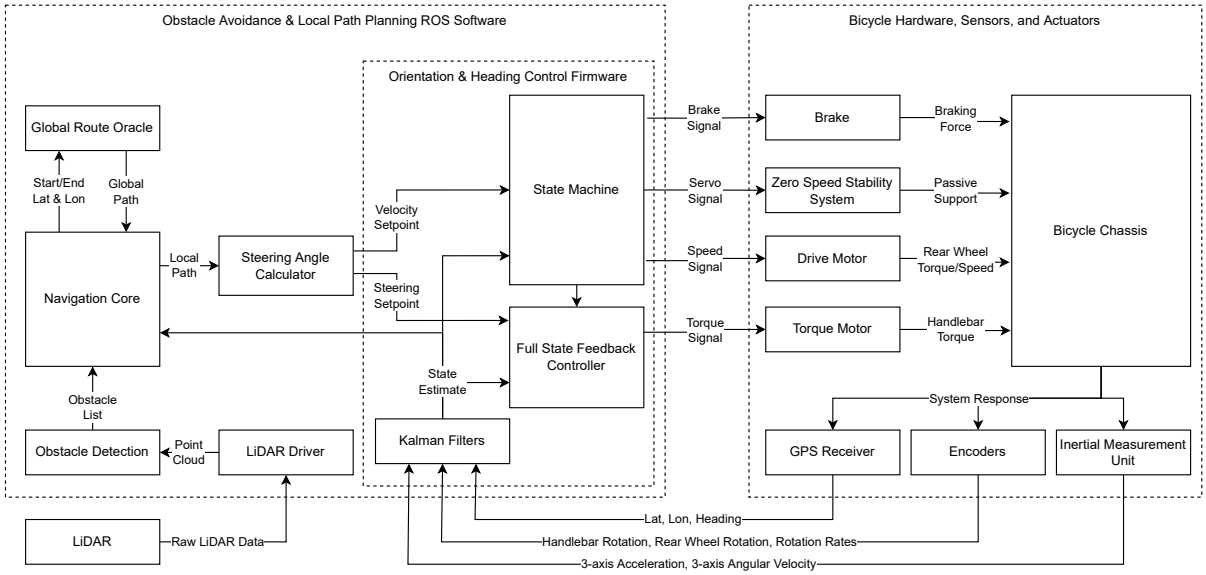


Figure 3.1: Simplified overall system block diagram.

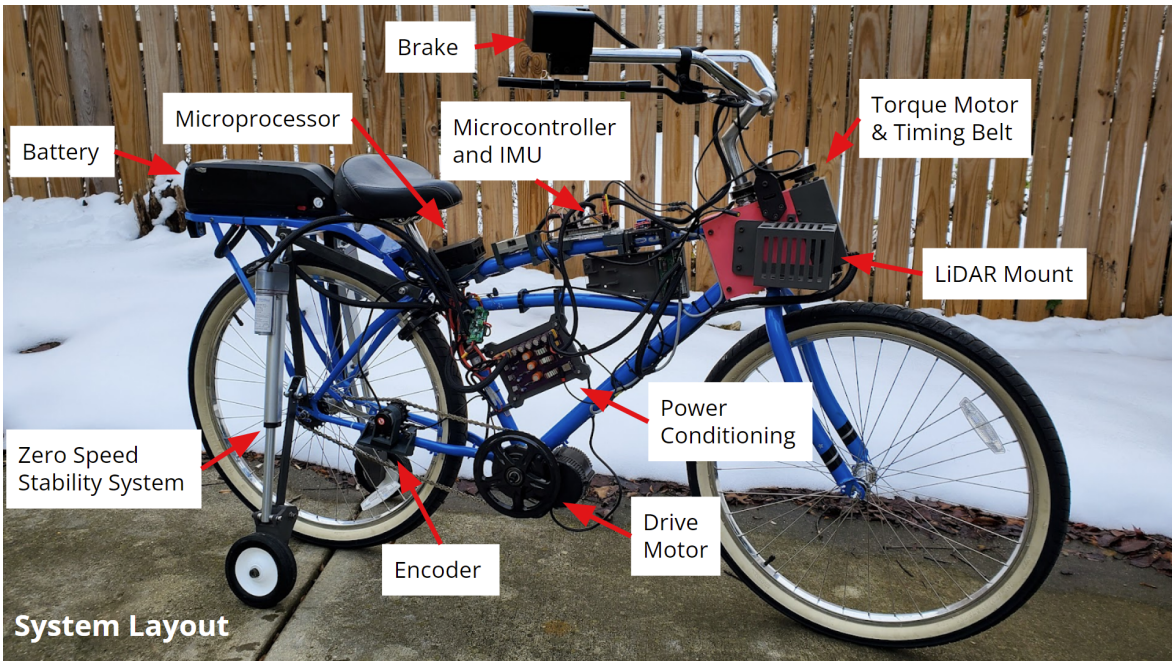


Figure 3.2: Overall system layout on the assembled Autocycle prototype.

3.1 Frame Component Rigid Bodies and Definitions

Our bicycle stability model simplifies all of the components of a bicycle into four rigid bodies. It is valuable to define the constituent parts of a bicycle, and the simplified bodies, to ensure clarity when discussing a stability model or actuation.

The four bodies used in the stability model are the front wheel, rear wheel, handlebar, and main frame. The front wheel is made of the rotating combination of the internal metal frame of the front wheel and rubber tire in contact with the ground. The rear wheel has the same components as the front wheel, but includes the rear sprocket, the gear that drives the wheel by way of a drive chain. The handlebar assembly is made of the handlebar, the metal structure that a human rider would twist to control the bike, the steering tube, defined as the tube running through the head tube of the frame, and the fork, the two prongs that connect to the front wheel's axle.

The main frame consists of the solid metal frame that makes up the majority of the bike, the seat, the seat post, pedals, and cranks.

The frame of our bicycle is made up of nine total tubes. The head tube is where the steering tube of the handlebar passes through the frame. It contains bearings which allow for smooth rotation of the handlebar assembly. The top tube is defined as the member in the frame that runs from the head tube to under the seat. The down tube is the member that runs from the head tube to the bottom bracket where the pedals are mounted. The seat tube runs vertically from underneath the seat to the bottom bracket. The top, down, and seat tubes make up the main triangle of a bicycle frame. The bicycle selected to serve as the frame for the Autocycle prototype also has a middle tube in between the top and down tubes. The seat stays are twin tubes that connect from the top of the seat tube to the rear axle. The two chain stays are tubes that connect from the bottom bracket to the rear axle. For a picture reference, see Figure 3.3 from [52].

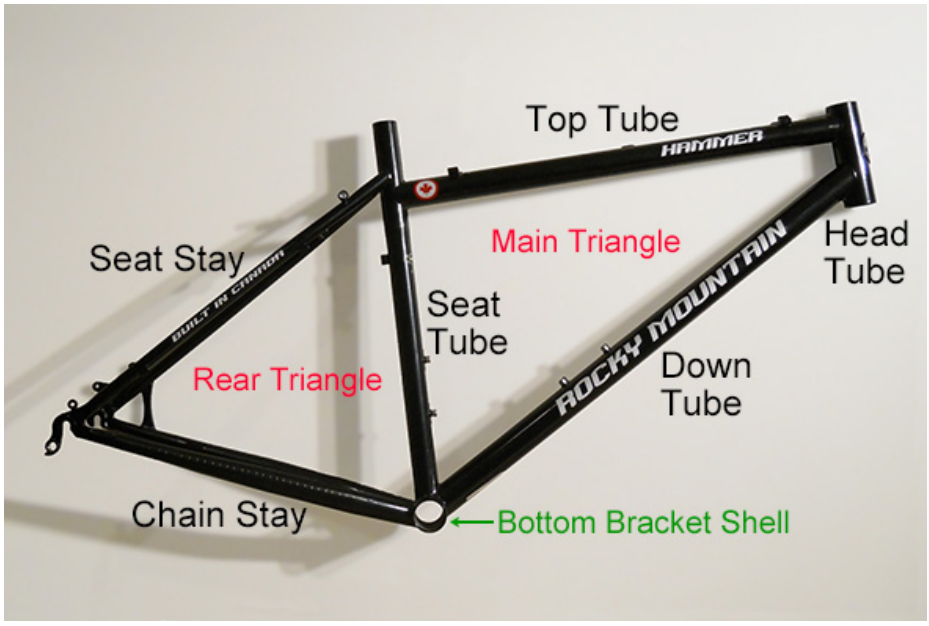


Figure 3.3: Labeled parts of bicycle frame

3.2 Characterization of Rigid Bodies

In order to mathematically model the bicycle's dynamics, knowledge of the bicycle's physical characteristics is required. At a minimum, values for the wheelbase, trail, and head angles; radius, mass, and moment of inertia of the wheels; and the mass, center of mass, and moment of inertia for the front and back frames are needed. To measure the components' weights, two separate methods were employed for redundancy. The first method involved the use of a digital personal scale. A member of the team first recorded their weight when standing on the scale alone. They then recorded the total weight when standing on the scale and holding one of the four rigid body components. Subtracting out the team member's recorded weight yielded the weight of the respective rigid body. This process was repeated for each of the four rigid bodies. In a second method, a string was tied to each rigid body and hung from a strain gauge. Both methods produced very similar results, and averaging the results gave weights of 3.05 kg for the front frame, 28.65 kg for the rear frame, 2.9 kg for the front wheel, and 3.3 kg for the rear wheel.

The locations of the center of gravity for each of the rigid bodies was determined as follows. Theoretically, the center of gravity is located along three axes for each rigid body. Each rigid body exhibited a degree of span-wise symmetry. Few methods exist to precisely locate the center of gravity along the rigid bodies' span-wise axis, so it was assumed that the center of gravity of each rigid body lies along its plane of symmetry. This reduced the number of axes along which to empirically locate the center of gravity to two. For each rigid body, a set of two perpendicular axes was defined. These axes are illustrated in figure 3.4. The procedure for measuring each rigid body's center of gravity is described below:

1. Suspend the rigid body from a string. Tie the string around a part of the rigid body that allows shifting the part along the first of the two defined axes.

2. Iteratively adjust the location of the string along the first axis until the part is evenly balanced. Etch the location where the string is tied to the part.

3. Repeat the same iterative process for the second of the two defined axes. Again, etch the final location where equilibrium is etched into the part.

4. Define one line from the first etching along the first defined axis and a second line from the second etching along the second defined axis. The center of gravity is the intersection of these two lines.

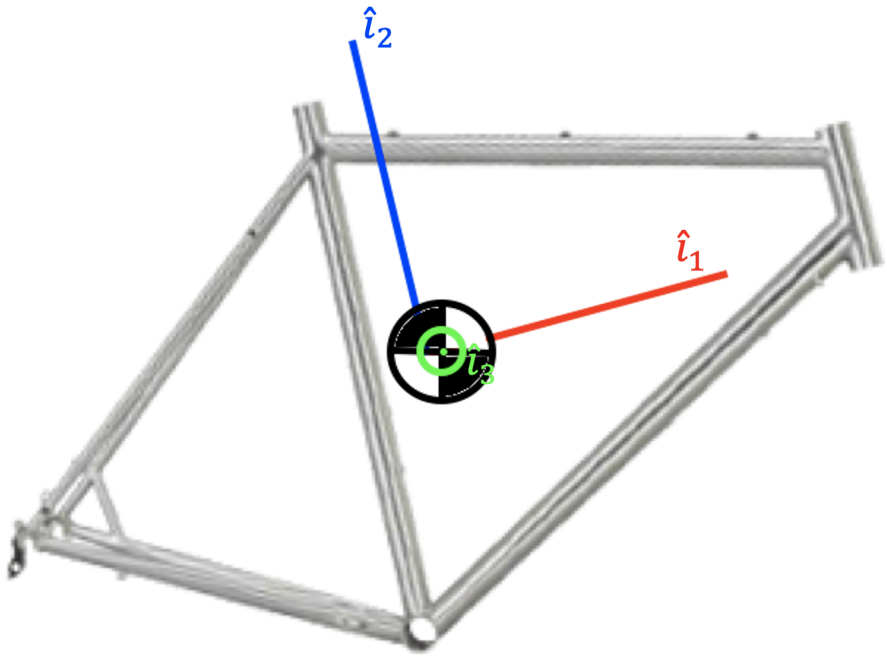


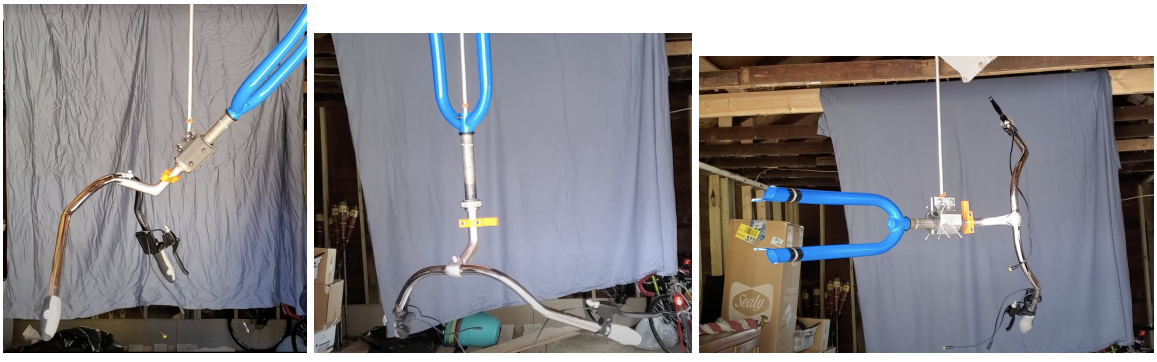
Figure 3.4: Example of coordinate axes system defined for main frame.

After identifying the locations of each rigid body's center of gravity, their respective mo-

ments of inertia (MOI) could be measured. The procedure for measuring MOI followed a torsional pendulum procedure described in similar studies of bicycle dynamics [16]. The team manufactured a test rig for hanging the rigid bodies from a 41.54 inch long, 0.25 inch diameter steel rod acting as a torsional pendulum. From these dimensions, the pendulum's MOI I_P was easily calculated. An aluminum clamp was fabricated by the team to hold parts ranging in shaft diameter at the plane of their center of gravity. For each part, an initial perturbation was applied to initiate periodic oscillation about the torsional pendulum. The time to complete 20 oscillations was measured for each part, in which the period for a single oscillation T could be deduced from simple calculation. The MOI was then calculated from Eq. (3.1).

$$I_M = \left(\frac{T}{2\pi}\right)^2 \left[\frac{GI_P}{l}\right] \tag{3.1}$$

This process was repeated for each part oriented along each of its three defined axes to measure I_{xx}, I_{yy}, and I_{zz}, as shown in Figure 3.5. These MOI values could then be used to compute each component's MOI tensor and define the bicycle model to predict its dynamics.



(a) I_{xx} (b) I_{yy} (c) I_{zz}

Figure 3.5: Handlebars mounted from torsional pendulum in three different orientations to measure I_{xx}, I_{yy}, and I_{zz}.

3.3 Power Conditioning and Distribution

The entire bicycle system uses a 48V nominal (13s) 12Ahr Lithium Polymer (LiPo) battery as our central power source. Many of the components required different voltages so we designed and build a custom power distribution board (PDB). The full voltage was sent to the Drive Motor, Torque Motor, and the Linear Actuators for the ZSS. Three step-down buck converters are used to output two 5V lines and one 12V line. The 12V line went to the Arduino Due and the Torque Motor. One of the 5V lines went to the LiDAR and another went to the Raspberry Pi. The regulators used have an output current limit of 5A which is why two 5V regulators were needed.

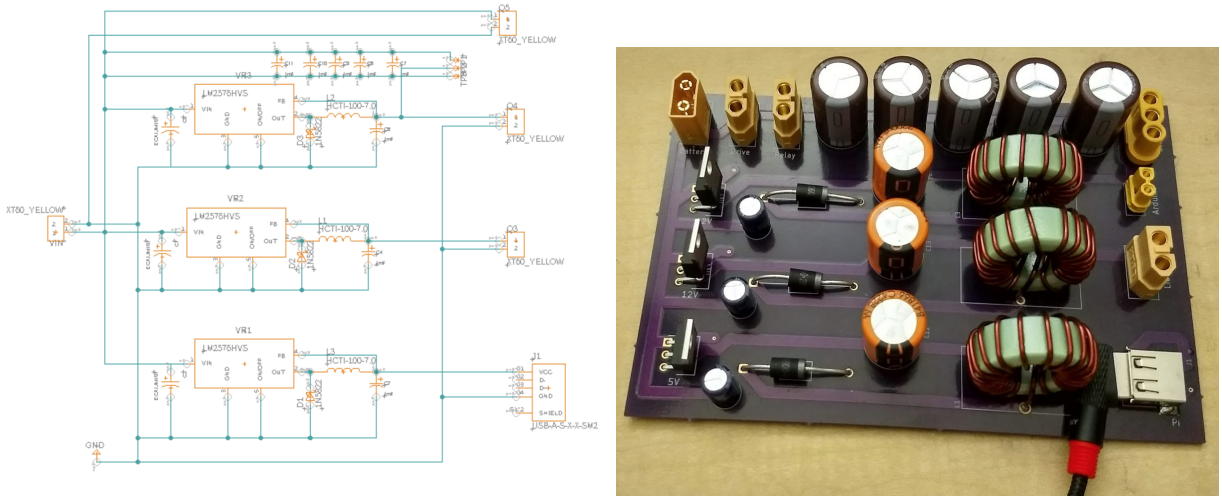


Figure 3.6: Design and Assembly of Our Power Distribution Board

Control boards can be damaged from large amounts of voltage ripple, so the design was made to limit the ripple to 20mV. Due to the size and weight of this project, the PDB design was not heavily restricted by weight or size. Large inductor coils were used to limit current and large through-hole capacitors were used due to their improved heat dissipation and the low application frequency.

3.4 Drive and Torque Motors

The main task of the bike is to move forward in a controlled manner towards its destination. The immediate accomplishment of this task is achieved using two actuators: the drive and steering torque motors.

3.4.1 Drive Motor

The bicycle was propelled by means of a 48V 750W commercial E-bike motor from Bafang, powered by a 13-cell Lithium Polymer battery with 12Ah capacity. The motor, like most E-bike conversions, was mounted by removing the old pedals from the bottom bracket and clamping the motor in their place. Due to the non-standard nature of the Schwinn Cosgrove Cruiser frame, this did require grinding the bracket's length down to fit the motor.

This sort of motor is typically used in pedal-assist mode, with a thumb-throttle (potentiometer) for additional acceleration. The speed is usually controlled by setting the pedal-assist (PA) mode via a small monitor which communicates with the motor over 1200 baud UART. The manufacturer does not provide the protocol used for communication between the monitor and motor, but thanks to prior efforts by hobbyists [53, 54] and a good amount of trial and error, the team was able to establish (limited) programmatic control of the motor's speed. A more complete description of the protocol is found in Section 3.8.2.

3.4.2 Steering Torque Motor and Belt Tensioner

Steering of the bike is accomplished by means of a Nanotec DB59L048035-B brushless DC motor, which, operating at 48V, can provide up to 0.47 N-m of torque. To increase the torque

output to necessary levels, a 32.72:1 reduction GP56-T2-33-HR planetary gearbox was used. The resulting output is applied to the handlebars via a timing belt system, using a 3/8in XL series belt. In order ensure proper torque transfer via the timing belts, keyslots were milled into the handlebar stem, and a tensioner was constructed. The motor was attached to the bicycle frame using a 3D-printed bracket, which also provided mounting points for the LiDAR and tensioner assemblies. Control was accomplished with CL4-E-2-12-5VDI motor controller.

The tensioner used a pair of idler pulleys mounted on shoulder bolts held in double shear in a pair of sliders. The sliders rode in a pair of dovetail grooves, with their position, and thus tension, controlled by a pair of 1/4-20 NC screws. The system was designed to be able to withstand torques much higher than those typically encountered in order to handle the very high transient loads that occurred during collisions or failed experiments. It also had the added effect of increasing the wrap angle for both the driving and driven pulley. The CAD of the entire torque motor/tensioner system can be seen in Fig. 3.7. The final iteration of the design was milled out of 1018 low-carbon steel and can be seen in in Fig. 3.8.

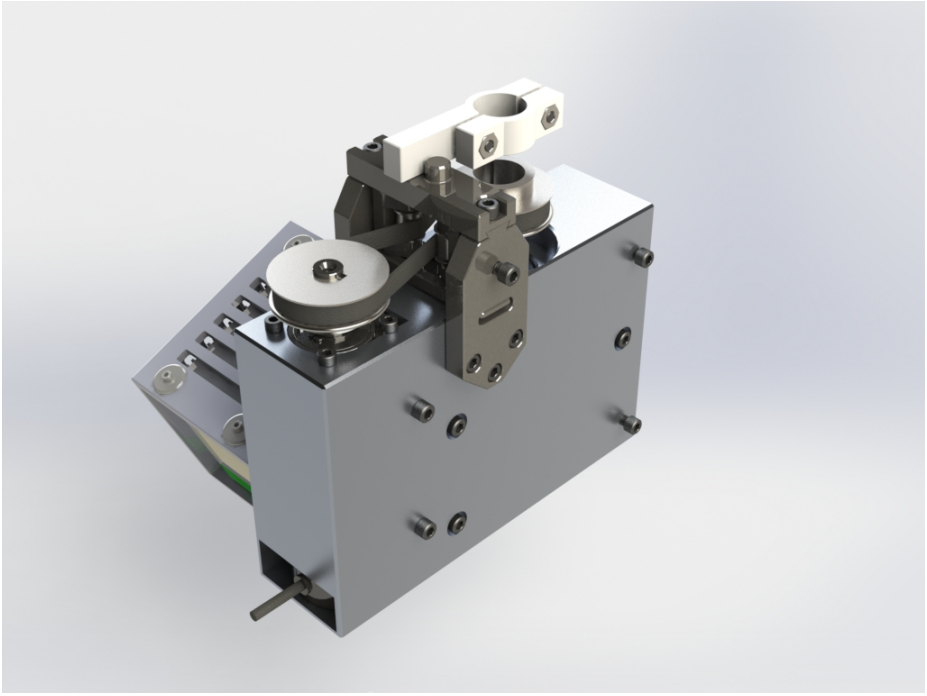


Figure 3.7: CAD of the torque motor mounting and belt tensioning systems.

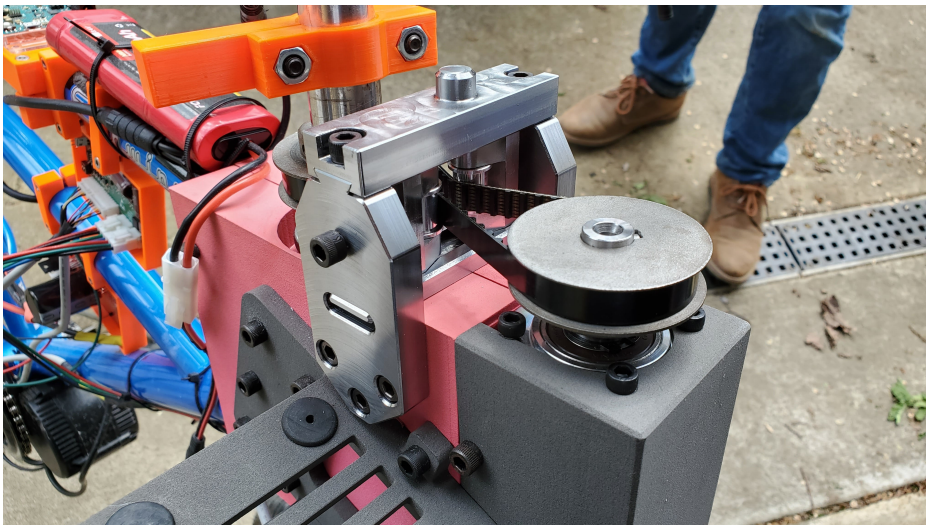


Figure 3.8: The assembled steel version of the tensioner and front wheel steering actuator.

3.5 Handlebar Brake Actuator

The bicycle must be able to slow itself down after it gets up to speed. The Autocycle, like other bikes, does this with a rim braking system, using a braking caliper attached to the rear wheel

which closes on the metal braking track, creating friction, and slowing the rotation of the wheel. The brake caliper is actuated by the brake lever, just as a rider would. A NEMA 17 stepper motor was used to actuate the break lever with a maximum output torque of 59 N-cm. Polymer fishing line was used to transfer the torque to the lever. At one end, the line was tied to the brake lever. At the other end, the line was wound around a 3D printed spool on the shaft of the motor. When the motor is actuated, the output shaft turns, which retracts the line and pulls the lever. In order to determine the required dimensions for mechanical advantage, an analytic optimization problem was solved to determine the optimal ratio of lever moment arm length to spool diameter. The spool was sized to be as small as it feasibly could be within our printing capabilities, and the calculated ratio was then used to determine the length that the lever arm needed to be extended. To extend the brake lever, a metal tube was attached to the end.

The motor was attached to the handlebars with a PETG 3D printed clamp. This was a viable material due to the small loads that would be applied to the mount itself, however due to the handlebar mounting location, the motor could impact the ground if the bike were to fall over. To mitigate any damage to the motor, an ablative shroud was printed as well. Multiple falls during testing showed that the shroud will often break during collision, but still successfully protects the motor. Multiple copies of the shroud were printed to allow for a swift replacement whenever the shroud sustained damage.

3.6 Zero Speed Stability System

One of the challenges that needed to be addressed early in development of the Autocycle prototype was stability when travelling at low speeds and in the event that the system needed to

come to an unexpected stop. The primary control input for maintaining stability, torque of the handlebars, is ineffective at stabilizing the prototype at very low speeds, and cannot stabilize the bike when it comes to a stop.

Mock-ups and conceptual sketches envisioned a system that utilized linear motors to extend or retract a pair of secondary wheels. The subsystem of actuators, wheels, and necessary structural support to attach its components to the bicycle would be referred to by the name "Zero Speed Stability System" (ZSS). This solution would allow the Autocycle to remain upright while motionless, and would help the system maintain stability when travelling at speeds too slow for control using handlebar torque. Moreover, should the Autocycle lean heavily to one side while running and fail to re-stabilize with the active control system, instead of falling to the ground and risking damage, the bicycle will fall onto the ZSS, allowing the subsystem to act as a safety net to prevent damage that would delay testing efforts or use. Once the Autocycle is travelling at a stable speed, the wheels retract off of the ground and the stability systems are solely responsible for stabilizing the Autocycle.

Figure 3.9: The first ZSS prototype in the ”deployed” configuration. The linear actuators on either side of the bike may raise or lower the stabilizing wheels by retracting or extending, respectively.

3.6.1 ZSS Linkage Design

The ZSS was integrated into the bike frame using fasteners and bolted joints. In the aft section of the Autocycle chassis, machine screws and threaded mounting points in the frame are used to attach the carriage over the rear wheel. These attachment points are fitted with longer screws of the same thread size that could mount steel support beams. The first support beam runs vertically, upward from the aft wheel's axis of rotation, where it connects to a horizontal steel beam that is fixed to the frame at an attachment point below the seat. The unattached ends of the frame are screwed together into an aluminum crossbar that connects both sides of the ZSS together underneath the aft carriage.

Figure 3.10: 1080 steel bars were selected for the ZSS linkage structure. The frame attaches to the chassis near the aft wheel axle, below the seat, and a threaded aluminum crossbar that both sides of the ZSS would affix to.

The steel frame provided a mounting point for pin joints between the moving parts of the ZSS and the chassis of the Autocycle. Brackets are bolted directly to the steel frame and fitted with shoulder bolts to allow for rotation of the members. Vibration and displacement due to loose fitting shoulder bolts inside the linking arm were minimized by mating a tight fitting bushing to the shoulder bolt and welding the bushing to the linkage arm. The linear actuator attaches to the constraining linkage arm and the support wheel by way of a 3D printed housing that bolts on to the extending end of the actuator and the end of the linkage arm. When extended fully, the support wheels contact the ground without pushing the aft wheel off of the ground. When the actuators are retracted, the wheels are lifted off of the ground and pulled outward to the sides of

the chassis, following the path constrained by the linkage arm.

Figure 3.11: When the bike reaches a stable travel speed, actuators retract the ZSS. This figure depicts the ZSS mid-retraction.

3.6.2 Linear Actuator Implementation

Linear actuators selected for implementation into the ZSS design needed to be robust and fast acting, with a stroke length that would allow the support wheels to ascend to a height that would not risk colliding with the ground when the Autocycle made tight turns. Progressive Automations' PA-15 High Speed Linear actuator was selected for its stroke length of 15.24 cm, fast retraction speed of 8.128 cm per second at no load, and high force output of 146.79 N ; it

was expected that if the bike was leaning to one side on the ZSS, it should output enough force to push the bike back upright.

The system underwent a redesign to increase the robustness of the ZSS control system, in which the actuators were replaced with new servos rather than simple DC linear actuators. Firgelli Automations' Feedback Rod Linear Actuators, which boasted better positional control than the PA-15's, were selected as the replacement for the old actuators following multiple hardware failures with the PA actuators.

3.7 Control and Sensing Unit

Real-time control of the Autocycle prototype was accomplished using a 32-bit Arduino Due microcontroller with a AT91SAM3X8E processor, running at 84 MHz clock speed [55]. The board was supplemented by a custom printed circuit board shield, which featured the following additional active components:

- 6-axis Inertial Measurement Unit (IMU) – MPU-6000
- 3.3v-5v Logic Level Shifter – LSF0204QPWRQ1
- Controller Area Network (CAN) Transceiver – SN65HVD230QD
- Stepper Motor Control – A4988
- Non-volatile Flash Storage – GD25Q16
- Radio Communication – NRF24L01P-R7

The IMU provided accelerometer and gyroscope data in three axes for orientation estimation. The logic level shifter allowed the 3.3V Arduino Due to communicate with the 5V drive motor

and encoder. The CAN transceiver allowed the Due to communicate with the front torque motor controller. The stepper motor controller drove the braking system. Non-volatile flash memory recorded data from each test. The 2.5GHz radio allowed wireless control and telemetry of the Autocycle during experiments. The shield schematic and final form are shown in Fig. 3.12 and 3.13 respectively. Additionally, a GPS unit (MTK3339) and magnetometer (LIS3MDL) were connected externally to provide data for position and heading estimation.

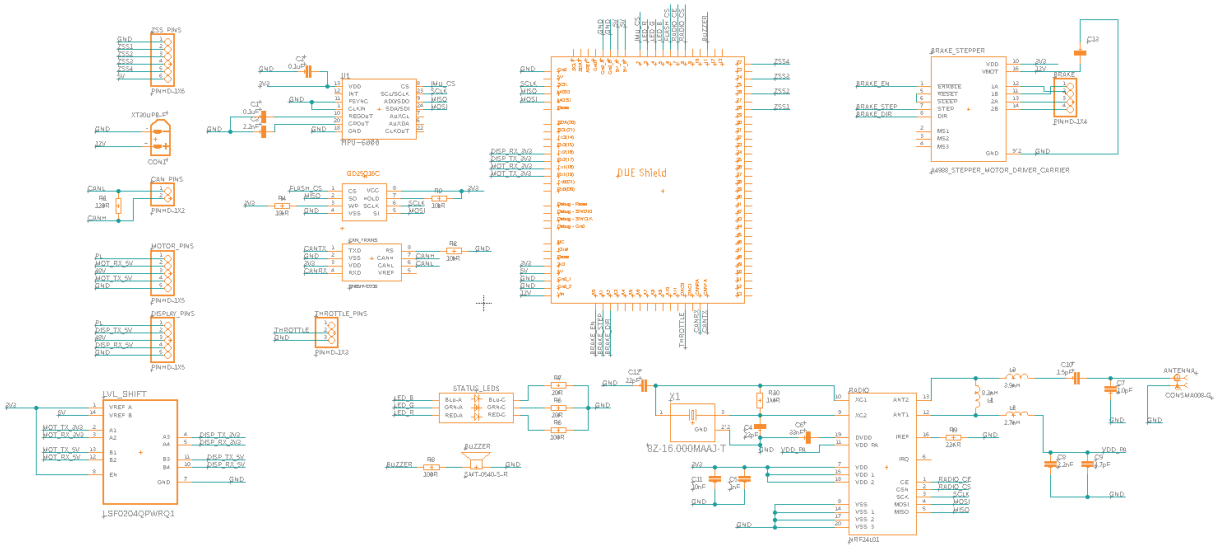


Figure 3.12: Wiring schematic of the control and sensing unit PCB shield.

The microcontroller receives commands and sends data by means of a 115200 baud emulated serial (UART) link over USB or over a 2.4GHz radio connection.

3.8 Firmware and Control

The Autocycle microcontroller runs firmware written in C++ using the open source Arduino framework. The firmware is tasked with keeping track of the system state, reading sensor inputs, estimating state variables, deciding on control outputs, and transmitting those outputs to

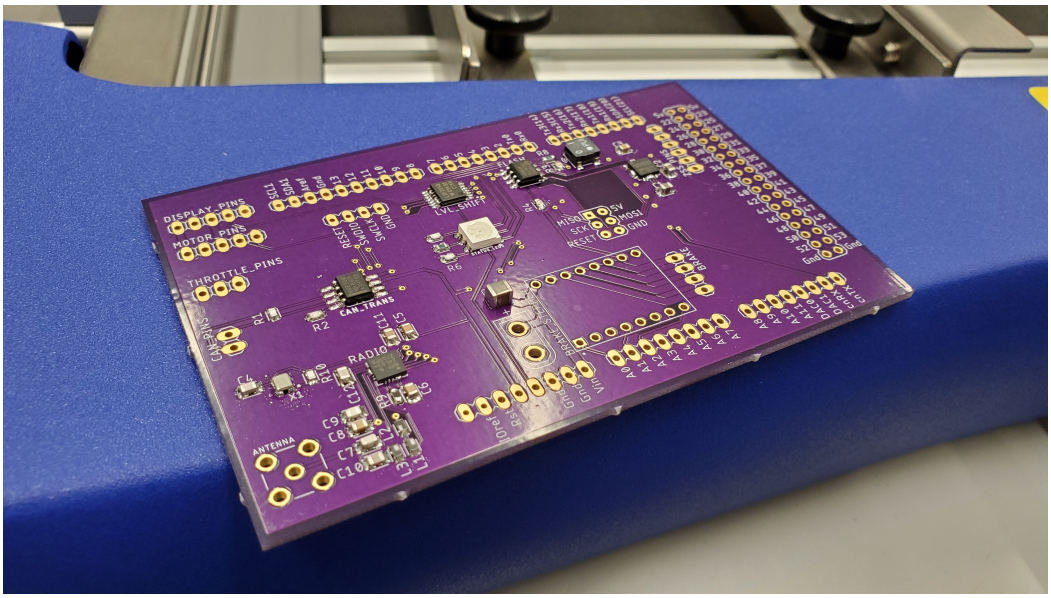


Figure 3.13: The final control and sensing unit PCB shield, with soldered surface mount components.

actuators.

The firmware architecture is structured in the "super-Loop" [56] real-time programming paradigm. Every iteration of the loop, the firmware performs the following tasks:

1. Calculate δt from previous iteration
2. Read sensor outputs using SPI, I2C, CAN, and GPIO interrupts.
3. Process sensor output using Kalman filter state estimation
4. Perform actions based on current machine state
5. Check for triggers for state transition
6. Repeat

3.8.1 State Machine

The firmware has to operate in several different modes in which different autonomous behaviour is required. A variable is maintained to track the current machine state, and the firmware behaves as a finite state machine.

Table 3.1: State machine states and descriptions

State No.	Name	Description
0	IDLE	With all motors off, await commands
1	CALIB	Await sensor calibration, do not respond to motion commands
2	MANUAL	Operate like a normal electric bicycle
3	ASSIST	Follow steering and speed commands with ZSS deployed (low speed)
4	AUTO	Follow steering, speed, and balance commands with ZSS retracted
5	FALLEN	With all motors off, await recovery
6	E_STOP	With all motors off and brakes on, await recovery

The various possible states are described in Table 3.1. Transitions between states are performed according to conditions illustrated in Fig. 3.14.

3.8.2 Sensor and Actuator Communication

Sensor data is read using a variety of methods. IMU data is collected over SPI in a batch. GPS data is retrieved using asynchronous UART. Data from the torque motor (position, speed, torque) is acquired over a CANopen Process Data Object (PDO) stream. Magnetometer data is retrieved over I2C.

The only sensor which is directly processed by the Arduino Due is the rear wheel optical encoder. The encoder provides a 5V quadrature waveform at 720 pulses per revolution. These pulses are passed through a 5 to 3.3 volt logic level shifter, and are then counted by means of a digital interrupt. In order to measure velocities, pulse counts are placed into 125 bins forming a

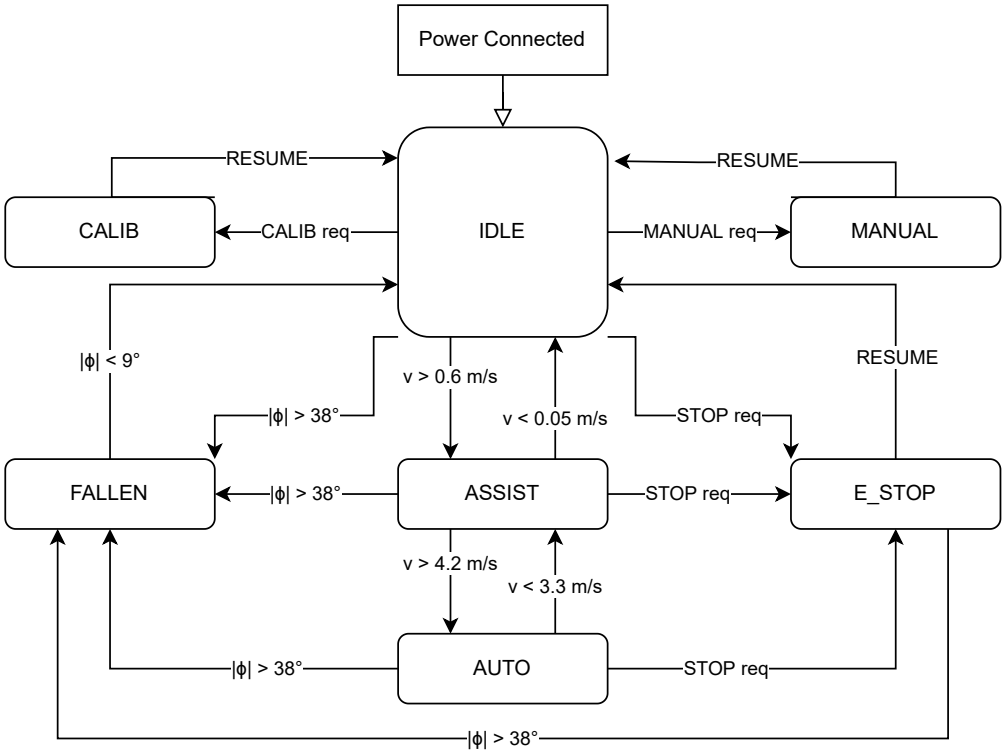


Figure 3.14: Autocycle firmware state transitions

circular array, with each bin representing pulses counted in a 2 millisecond interval. The current active bin is updated based on the onboard clock, clearing the next bin. Whenever a velocity measurement is needed, the bins are summed to get the instantaneous count of pulses in the past 250 milliseconds, divided by .250 to get pulses per second, and multiplied by a conversion factor representing sprocket tooth ratios and wheel circumference to arrive at a measurement in meters per second.

Communication with the Nanotec steering torque motor controller is accomplished using the CANopen network protocol over a 1 Mb/s CAN bus link. Control parameters are set using Service Data Objects (SDOs), while real-time torque, position, and speed setpoints are specified using Process Data Objects (PDOs). The specific register map for a CL4-E-2-12-5VDI controller is described in full in the device manual [57].

Communication with the Bafang drive motor is much more primitive, using a 1200 baud UART connection to set the various Pedal Assist Mode speeds and in order to switch between them. The specific protocol for the drive motor was not documented, and so needed to be reconstructed from extant online experimentation by hobbyists [53, 54] and trial and error. On/off throttle is achieved with a 3.3V signal connected to the sense line of what would normally be the thumb-throttle potentiometer. Since the motor's logic operates at 5V, a logic level shifter was used on the UART lines.

Control over the ZSS actuators is accomplished with a simple 5V RC servo PWM signal sent to the actuator controller through a voltage level shifter. The A4988 stepper motor driver used for the braking system simply receives a direction signal and individual step pulses from the control and sensing unit.

3.8.3 Control Design

With the requisite hardware in place, it is necessary to determine the control law and gains that will be implemented. For the Autocycle prototype, we chose to proceed using the fourth order linear model described in 2.1. with the only control input being the torque on the handlebars for steering. This model makes several major assumptions that should be addressed. First, as it is derived from the Whipple model, it assumes rigid bodies, frictionless rolling joints, and non-slip non-holonomic knife-edge contacts between the wheels and the ground. Second, as part of the linearization, the forward velocity of the bicycle (as measured at the contact point of the rear wheel) is assumed to be constant. This assumption is clearly not met during the acceleration of a bicycle from rest, and assuming velocity to be constant removes a potential input/actuator that

could be used for control.

Nevertheless, we chose to proceed with this model for several reasons. First, experimental work had been carried out [16] validating the linearized model compared to empirical measurements on an uncontrolled bicycle. Second, the model's linearity allowed for the use of powerful but simple classical and modern control methods. Third, while the velocity was set by an actuator we had nominal control over (the Bafang drive motor), we were only able to set the velocity in very coarse increments, often with a large steady state error. Thus, it made more sense to treat the speed as a parameter constant in the short term, and to simply measure speed precisely instead of attempting to control it precisely.

Additionally, as mentioned in Sec. 2.7, multi-input control schemes have been proposed or tried in the past, usually including an actuator, such as reaction wheel or control moment gyroscope, to impart a torque directly on the roll axis. While such actuators clearly could maintain the orientation of a bicycle, having essentially reduced the problem to an inverted pendulum, the steady state power draw, potential for saturation, and undesirable precession effects during turning led to the decision to use only steering torque for control.

3.8.3.1 State Space Model and Initial Parameter Estimates

To arrive at the initial system model for design, the various lengths, angles, masses, and moments of inertia were measured using the procedure described in Sec. 3.2. This gave the mass, stiffness, and damping matrices for the 2-dimensional, 2nd order linear model. From our measurements, the matrices were as follows:

$$M = \begin{bmatrix} 18.7039325 & 0.63172415 \\ 0.63172415 & 0.39746713 \end{bmatrix} \qquad C_1 = \begin{bmatrix} 0. & 11.05093702 \\ -1.13153138 & 0.97882643 \end{bmatrix}$$

$$K_0 = \begin{bmatrix} -244.351404 & -9.65862573 \\ -9.65862573 & -2.40638906 \end{bmatrix} \qquad K_2 = \begin{bmatrix} 0. & 22.48449891 \\ 0. & 1.03238589 \end{bmatrix}$$

However, to proceed with standard state-space tools, it was necessary to transform this into a 4-dimensional, 1st order linear model. The derivation was performed as follows:

$$M\ddot{q} + vC_1\dot{q} + (K_0 + v^2K_2)q = g$$

Solving for $\ddot{q} = M^{-1}(g - cC_1\dot{q} - (K_0 + v^2K_2)q)$

$$\text{Let } x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \text{ Let } u = g$$

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1}(g - cC_1\dot{q} - (K_0 + v^2K_2)q) \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} 0 & I \\ -M^{-1}(K_0 + v^2K_2) & -vM^{-1}C_1 \end{bmatrix} x + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} u$$

This puts the system in state space form, with

$$\dot{x} = Ax + Bu$$

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}(K_0 + v^2 K_2) & -vM^{-1}C_1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}$$

3.8.3.2 Open Loop Dynamics and Controllability Analysis

Having identified the "static" parameters of our model, it is necessary to inspect the expected open loop dynamics would be for a range of velocities. We could do this by calculating the full A and B matrices for a given velocity, then finding their eigenvalues. A plot of the real parts of the eigenvalues vs. speed is shown in Fig. 3.15.

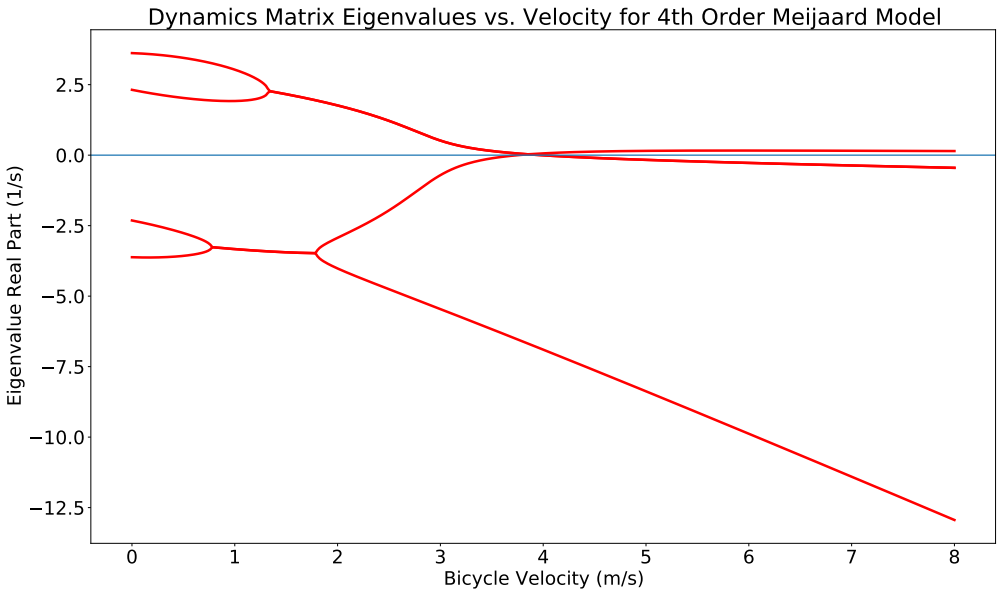


Figure 3.15: Evolution of the real part of the system eigenvalues with bicycle velocity for the Autocycle prototype chassis.

As the figure shows, the system is unstable, with one or more eigenvalue real parts being

positive for almost the entire operating range considered. The system only approaches marginal stability in a very narrow band around 4 m/s. This response could be changed by altering the geometry and mass distribution of the bicycle, making it stable over a greater range of speeds. However, this would at best make the system capable of regulating itself back to zero roll and steering angle (upright and front wheel pointed forwards), but not make it capable of going anywhere in particular.

Thus, the system requires active control, and so we must check whether it is controllable. Programmatically constructing the controllability matrix for a 4th order system, $R = [B\ AB\ A^2B\ A^3B]$ and finding its rank for a range of speeds shows that the matrix R is full rank for all nonzero bicycle speeds, and thus that the (linearized) system is controllable for all such conditions.

3.8.3.3 State Space Orientation Regulation

Having found that the system was unstable, but controllable, control of the orientation (roll and steering angles) of the bicycle was accomplished using a full state feedback controller, that is,

$$u = -Kx$$

Which gives effective closed loop system dynamics,

$$\dot{x} = (A - BK)x$$

Also known as pole placement, this control law allows us to choose gains to arbitrarily pick the closed loop eigenvalues of the system. We chose to place the eigenvalues at -6, -7, -8, -9 with the aim of achieving relatively fast response without oscillation.

Since the linearized system dynamics were specified to be dependent on velocity, the system model was updated in real-time with the measured velocity on every iteration, and new corresponding gains were computed. This could be considered an implicitly Linear Time Varying system.

Since the controller was designed for a continuous system model, in implementation on a discrete time software, it was susceptible to misbehaving if iterations were not run at sufficiently fast intervals, with long zero-order holds resulting in instability. Eliminating such errors was a matter of proper time management in the firmware architecture.

3.8.3.4 Orientation Setpoint Tracking

While the controller described above succeeded at regulating the system state to near the origin (upright motion with zero steering angle), it was also necessary for the Autocycle to be able to execute deliberate steering commands. To this end, we need to find other stable equilibria beside the origin to which we would then regulate the system. To find these equilibria, we can consider the second order, 2-dimensional model as described in Section 2.1. To find equilibria, we set the first and second derivatives of orientation to zero, and solve for the torque and roll for a given steering angle, as is done in the derivation below:

$$M\ddot{q} + vC_1\dot{q} + (K_0 + v^2K_2)q = g$$

$$\ddot{q} = 0, \dot{q} = 0$$

$$(K_0 + v^2 K_2)q = g$$

$$(K_0 + v^2 K_2) \begin{bmatrix} \phi_r \\ \\ \delta_r \end{bmatrix} = \begin{bmatrix} 0 \\ \\ T_{\delta_r} \end{bmatrix}$$

Since we would like to control the system with a steering setpoint, we will now proceed to solve for equilibrium roll and torque as a function of equilibrium steering.

$$\text{Let } \bar{K} = (K_0 + v^2 K_2) \text{ with } k_{i,j} = \bar{K}_{i,j}$$

$$\text{Then, } \begin{bmatrix} k_{1,1} & k_{1,2} \\ \\ k_{2,1} & k_{2,2} \end{bmatrix} \begin{bmatrix} \phi_r \\ \\ \delta_r \end{bmatrix} = \begin{bmatrix} 0 \\ \\ T_{\delta_r} \end{bmatrix}$$

$$k_{1,1}\phi_r + k_{1,2}\delta_r = 0$$

$$k_{2,1}\phi_r + k_{2,2}\delta_r = T_{\delta_r}$$

$$\phi_r = -\frac{k_{1,2}}{k_{1,1}}\delta_r$$

$$T_{\delta_r} = (k_{2,2} - \frac{k_{1,2}k_{2,2}}{k_{1,1}})\delta_r$$

Of course, these are the equilibria of the linearized system, which itself was constructed by linearizing the true model dynamics about the zero steering/zero roll equilibrium. As such, this choice is not ideal, but given the unwieldiness of the full nonlinear governing equations, we

chose to proceed.

Having found these non-zero equilibria, we can modify our control system to include a steering setpoint. To do this, we perform regulation with the aforementioned full state feedback gains on the difference between the current and reference state, while adding the equilibrium torque found above.

$$\dot{x} = Ax + Bu$$

$$x_r = \begin{bmatrix} \phi_r \\ \\ \delta_r \end{bmatrix}$$

$$\delta x = x - x_r$$

$$u = -K\delta x + \begin{bmatrix} k_{2,1} & k_{2,2} \end{bmatrix} x_r$$

So the resultant dynamics are

$$\dot{x} = Ax + B(-K(x - x_r) + Ex_r)$$

$$\dot{x} = (A - BK)x + (BK + E)x_r$$

$$\text{for } E = \begin{bmatrix} k_{2,1} & k_{2,2} \end{bmatrix}$$

3.8.3.5 Heading Control

After the orientation control was shown to stabilize the bicycle, heading control was accomplished using a simple proportional controller that worked on the current estimated heading as the measured variable and used the steering setpoint into the orientation controller as the control variable.

Design was performed by constructing the transfer function between steering setpoint and heading angle. To find the needed transfer function, we can consider the system above, with the steering angle δ as our only output (that is, $C = [0\ 1\ 0\ 0]$). This results in the following:

$$\dot{x} = (A - BK)x + (BK + E)x_r, y = Cx = \delta$$

$$x_r = \begin{bmatrix} -k_{1,2} \\ \\ 1 \end{bmatrix} \delta_r$$

$$\text{Let } A_c = (A - BK), B_c = (BK + E) \begin{bmatrix} -k_{1,2} \\ \\ 1 \end{bmatrix}$$

$$\dot{x} = A_c x + B_c \delta_r$$

Applying the Laplace transform,

$$sX(s) = A_c X(s) + B_c \Delta_r(s)$$

$$(sI - A_c)X(s) = B_c \Delta_r(s)$$

$$X(s) = (sI - A_c)^{-1} B_c \Delta_r(s)$$

$$\Delta(s) = C(sI - A_c)^{-1}B_c\Delta_r(s)$$

$$\frac{\Delta(s)}{\Delta_r(s)} = C(sI - A_c)^{-1}B_c$$

A common steady state relation between steering and turning radius for two-wheeled vehicles is given by [58],

$$\delta = \frac{L}{R} + (\frac{m_f}{C_{\alpha f}} - \frac{m_r}{C_{\alpha r}})\frac{v^2}{R}$$

For wheelbase L, turn radius R, front and rear wheel masses m_f and m_r, and front and rear tire cornering stiffnesses $C_{\alpha f}$ and $C_{\alpha r}$. Since the turning radius is related to the speed and heading rate by $v = R\dot{\theta}$ for θ heading, it follows that

$$\delta = c\dot{\theta}, \text{ for } c = \frac{L}{v} + (\frac{m_f}{C_{\alpha f}} - \frac{m_r}{C_{\alpha r}})v$$

$$\Delta(s) = sc\Theta(s)$$

Substituting into the steering/steering setpoint transfer function, this gives

$$\frac{\Theta(s)}{\Delta_r(s)} = \frac{1}{cs}C(sI - A_c)^{-1}B_c$$

This transfer function was then used to select a proportional controller with desired characteristics using a root locus plot (See Fig. 3.16). It was found that gains surpassing 0.2 would lead to instability. A gain of 0.1 was found to give a rise time of 1.5 seconds with no overshoot, mild undershoot, and reasonable phase and gain margins, and this gain was deployed on the Autocycle.

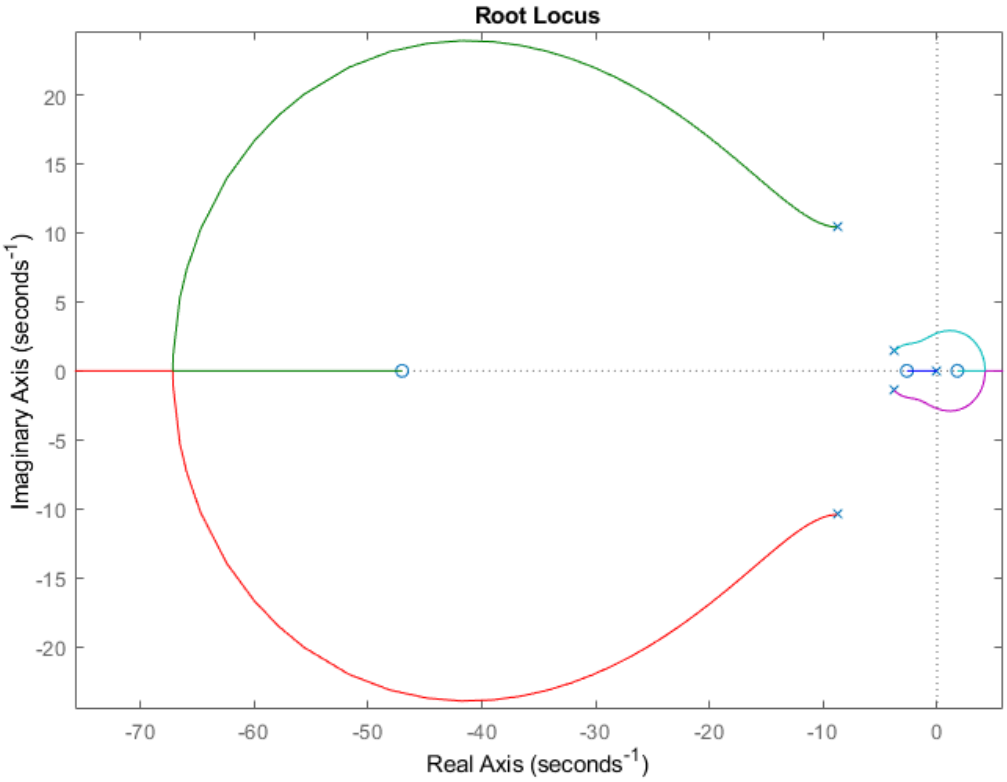


Figure 3.16: Root locus plot of heading control via proportional feedback

3.8.3.6 System Identification

Initial testing using full state feedback control based on the model parameters found via measurement of moments of inertia proved unsuccessful in stabilizing the system. In order to improve the quality of control, it was necessary to improve the quality of the system model and parameters.

A significant problem faced in this system identification problem is that one of the state variables, ϕ, can not be measured directly during experimental operation of the bicycle. While orientation can be measured at rest by means of accelerometers, when the bicycle is moving far too many extraneous accelerations make this measurement too noisy and uncertain to be useful.

As such, for the purposes of control ϕ is estimated by means of a Kalman filter, largely drawing on the ϕ rate information obtainable from onboard accelerometers. As the Kalman filter is inherently dependent on the system model used during the experiment, it did not seem to be a reliable state variable to perform SYSID on.

As such, the ϕ and δ rates were used for the first round of fitting. However, the evolution of the ϕ and δ rates depends directly on ϕ and δ themselves. Thus, if the recorded ϕ could not be used, 1-step ahead prediction was not possible, and so simulation, aka infinite-step ahead prediction had to be used.

System identification was performed in Python using the NumPy and SciPy scientific computing packages for matrix operations and optimization tools. While the goal was to estimate the values of the four 2x2 matrix M, C_1, K_0, and K_2, only 11 parameters were needed (instead of 16) due to certain assumptions about symmetry or certain elements being 0 that could be made [59].

Optimization was initially performed using the `scipy.optimize.minimize` general nonlinear minimization routine, using the Nelder-Mead method, with the objective function being the sum of squared errors between the actual roll and steering rates recorded in the data. However, with 11 degrees of freedom, while even the earliest models significantly outperformed the naive model, they sufferred significantly from overfitting.

A major component of this effect was that in the simulation, the inverse of the mass matrix is multiplied by the total stiffness and damping matrices. As such, the mass matrix could vanish provided that the stiffness matrix approached a large value and vice versa while still arriving at roughly the same simulation results.

In order to combat this, a regularization term was introduced into the the objective function for the minimization, penalizing excessive deviation from the the "naive" model, thus making the

total objective function

$$J(\theta) = \frac{1}{N}\sum_{i=0}^{N} |\dot{\hat{q}}(i;\theta) - \dot{q}(i)|^2 + |\theta - \theta_0|$$

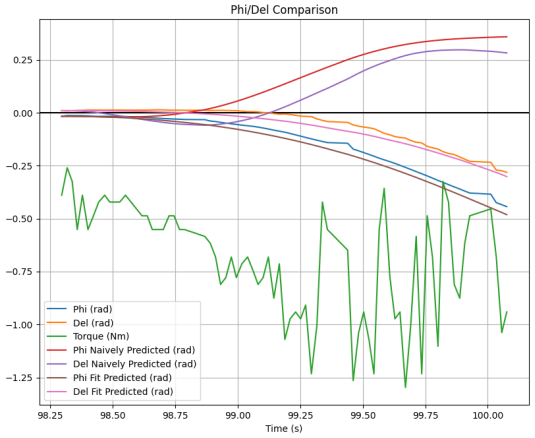


Figure 3.17: Data and simulated time evolution of the roll angle and steering angle.

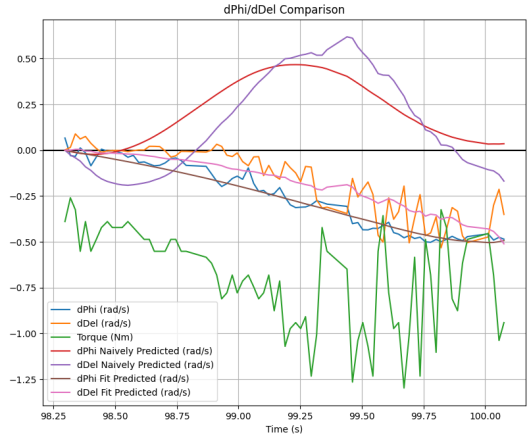


Figure 3.18: Data and simulated time evolution of the roll and steering angle rates.

With this term added, parameters converged to much more reasonable values. The simulated state trajectories using these new parameters were then plotted against experimental data as seen in Figures 3.17 and 3.18.

An inherent difficulty with the fitting of these parameters was that despite the model assumption that the velocity remained constant, it in fact did vary to some extent, even during the reduced window of the data used for parameter fitting. In addition, the open loop system is unstable, and so simulated results should tend to diverge from true behaviour exponentially for any nonzero initial error in the state estimate. Thus, the fitted parameters were tenuous, but appeared to work well.

3.8.4 State Estimation

State estimation is performed using a set of three Kalman filters, which provide optimal state estimation for linear systems with Gaussian noise and known covariances [60]. For each of these filters, transition and control matrices are found by taking the 1st-order Euler discretization of the dynamics. Namely, for a dynamical system $\dot{x} = Ax + Bu$, the discrete dynamics are approximated as $x_{k+1} = A_d x_k + B_d u_k$ for $A_d = I + A\delta t$ and $B_d = B\delta t$. This assumes zero order holds for control inputs.

3.8.4.1 Orientation Estimation

Orientation estimation (roll, steering angle, roll and steering rates) is accomplished using X and Y accelerometer, X gyroscope, and encoder data and the linearized equations of motion described in Sec. 2.1 and their conversion to state-space form described in Sec. 3.8.3.3. While the MEMS rate gyroscopes provide a direct measurement of the roll rate (after compensating for bias and IMU rotation relative to the system axes), the accelerometers can be used to provide an absolute (albeit noisy) measurement of the roll by taking the arctangent of the Y and X

components of the acceleration vector.

In order to prevent estimate drift (and due to the high resolution of the steering motor's incremental encoder), the filter is used as a partial state estimator, with steering angle and steering rate being taken as truth, and roll angle and roll rate being estimated. Given that the dynamics for theses states changed depending on if the ZSS is deployed or not, the firmware Kalman filter takes this into account and decouples the roll and steering dynamics if the training wheels are on the ground.

3.8.4.2 Heading Estimation

Heading estimation is accomplished using the GPS heading data and the Z gyroscope's reading. When the bicycle is in motion, the GPS module is able to estimate heading by taking finite differences of positions, providing an absolute measurement. At the same time, the gyroscope provides a measurement of the heading rate. The filter uses simple planar rigid body rotation dynamics.

3.8.4.3 Position Estimation

Position estimation is accomplished by combining GPS latitude and longitude readings with the bicycle's speed (as given by the rear wheel encoder) and heading. GPS readings provide an absolute measurement of position at a low refresh rate, while the known speed is converted to a planar velocity vector with latitude and longitude components. While the meter/second to latitude conversion remains constant, the meter/second to longitude conversion changes based on the previously estimated latitude to account for the narrowing between the lines of longitude as

one approaches the poles. Simple planar point kinematics are used for the dynamics matrix.

3.8.4.4 Filter Tuning

Sensor covariances for each of the aforementioned filters where measured empirically. This left the process covariances an available tuning knob. To achieve optimal orientation estimation, filter process covariances were tuned with respect to a ground truth. This was accomplished by using a reference rotation jig, as shown in Fig. 3.19. A stepper motor was used to rotate the control and sensing unit in a known oscillatory fashion. The on-board roll and roll rate estimate was recorded, and process covariance gains were adjusted to minimize the mean squared error between the known roll trajectory and the system estimate.

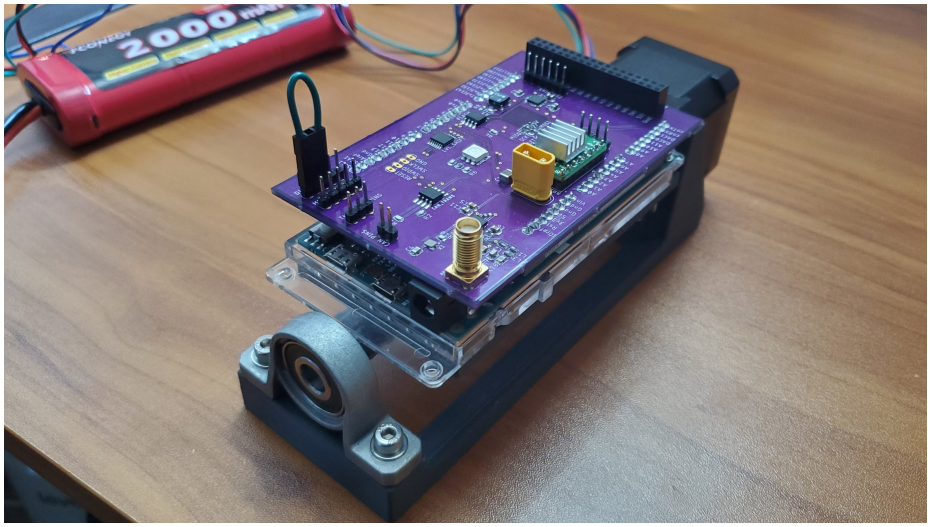


Figure 3.19: Experimental system for orientation Kalman filter tuning.

3.9 LiDAR

Central to the navigation systems on board the bicycle is the ability to detect obstacles in its path. In order for any system to do so, that system must have some way of scanning its

environment. As stated in the literature review, this scanning can be performed by a variety of systems including cameras, RADAR, SONAR, LiDAR, etc. From these options, LiDAR was selected. LiDAR, when used in its rated environments, provides highly-precise point clouds containing points of reflectively in the environment. The points in this cloud correspond to points where there may be an obstacle that must be taken into consideration.

For this project the Livox Horizon was selected for its relatively low price and sufficiently large viewing range. Though other better LiDAR options existed at the time, they were either too bulky, outside of the available budget, or difficult for a student research team to acquire. The selected LiDAR provides a 260 m detection range, 81.7° × 25.1° horizontal x vertical FOV, 2 cm range precision, 0.05° angular precision and a data rate of 240,000 points per second. Each point of data provided by the LiDAR is composed by the (x,y,z) position of the point relative to the LiDAR's position as well as a value describing the reflectivity at that point in space. In addition to this data, the LiDAR periodically records angular velocity and acceleration via an on board inertial measurement unit.

The LiDAR is able to communicate with other computational systems using a Software Development Kit (SDK) provided by Livox. In its default method of operation, the SDK is set to run while outputting its data continuously into an output file. For use in this project the SDK was modified such that the output of a single full frame is recorded to a Linux socket at a time. That frame is then read in by other systems on the bicycle and the Livox SDK is directed to record a new frame via the same socket.

3.10 Navigation Computational Unit

To process the LiDAR point cloud and to run the local and global route planning algorithms described in the following section, a sufficiently powerful computational unit is required. There are limitless options when it comes to computational units including GPU/CPU embedded systems, cellular devices, single board computers, etc. Due largely to budgetary constraints, a single board computer, the Raspberry Pi 4B, was selected. This model runs a 4-core 64 bit SoC at 1.5 GHz and 8 GB of SDRAM.

3.11 ROS Powered Navigation

The bicycle's navigation system consists of several intercommunicating processes that navigate the bicycle around obstacles and to a desired destination. This section covers the different subsystems that power these Navigation capabilities.

3.11.1 Robot Operating System

The navigation system for the bicycle requires a number of synchronously operating programs that must be able to communicate with one another. Solving this communication problem can prove a rather daunting task. One way to do so is using Linux sockets as done with the LiDAR SDK. A second more effective way however is to leverage existing tools in the Robot Operating System (ROS).

ROS allows for multiple programs to run synchronously as ROS nodes. Nodes are able to communicate with each other either trough messages or through services. In the case of messages,

a node can *publish* a *message* to a named *topic*. Any node can also *subscribe* to a named topic and read any message that get published to that topic. Because multiple nodes can subscribe to the same topic, this is effectively a one-to-many communication scheme where direct interaction between nodes is not required. Services, on the other hand, are functions or methods provided by a node that can be called by another node. When a node that implements a service is started, it advertises that service to all other nodes. Another node can then use the service; in doing so, it often sends some data and receives some resulting output. Unlike messages this results in direct one-to-one messaging between nodes. Services also allow for a two-way exchange between nodes without having to use multiple messages. Crucially, with a service, the calling node will hang until the service provided is complete. With this in mind, the ROS architecture in 3.20 describes the way nodes in the navigation pipeline communicate with one another. Though the details of what each node does will be explained further, below is a brief overview of what each node does.

- Navigation Core : Contains the majority of navigation computation. Route planning, object detection, point cloud parsing etc...
- LiDAR Driver : Starts the LiDAR and informs the navigation core when there is a new frame to process
- Start Data : Starts the Navigation Core's data processing
- Global Route Oracle : Provides desired (latitude, longitude) values to the Navigation Core upon query
- Path Interpolation : Uses b-splines to interpolate the path generated by the navigation core

- Provide Steering Angle : Uses interpolations from the previous node to generate desired steering angle throughout the path
- Serial Write : Starts the bike's movement. Queries the Steering Angle Provider for a steering angle given a current bike position and sends that to the Arduino Due.
- Serial Read : Reads telemetry data from the Arduino Due and sends that data to the requisite nodes. Also stalls Navigation Core and Serial Write until serial connection is operational.
- Start Movement : Stalls Serial Write from starting the bike movement until this node is started.
- Retract ZSS : Retracts the ZSS actuators
- Deploy ZSS : Deploys ZSS actuators

All nodes with the exception of *Start Movement*, *Start Data*, *Retract ZSS* and *Deploy ZSS* are started automatically. *Start Movement*, *Start Data* are manually started to control when the bicycle performs the related actions. *Retract ZSS* and *Deploy ZSS* are also started manually whenever the ZSS is not in the appropriate position prior to a test.

3.11.2 Global Route Planning

There is little point in having the bike move and avoid obstacles if there is no set destination. For the sake of this section, local route planning can be thought of as a black box that brings the bicycle from some starting position to some end position within 20 meters or so. In order for the bicycle to get further, a series of end positions needs to be provided to the local route planning

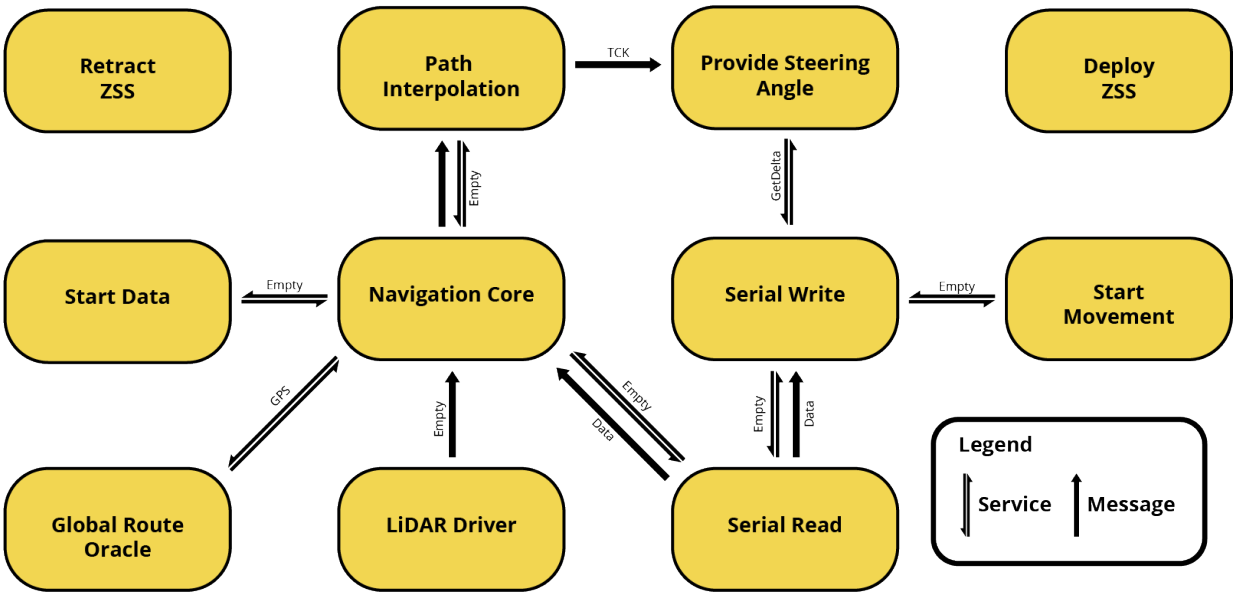


Figure 3.20: ROS architecture used to run navigation systems for bicycle. Each rectangle represents a ROS node. One directional arrows indicate a publisher-subscriber relationship. Bidirectional arrows describe servicer-beneficiary relationship. Labels on the arrows describe the datatype being sent. Arrows with label Empty indicate a relationship where the only goal of the relationship is to indicate some change in state. For example, the Navigation Core waits until the Start Data service starts before continuing, but it never actually requires any further information. In addition to these nodes there are other nodes that always run in a ROS architecture like *roscore* and *out*, but those are excluded for legibility.

black box. Global route planning aims to generate a series of positions that eventually lead to the final desired end position. The local path planning black box can then query the global route planning for subsequent destination positions. In this way, the global route planning can be considered as an oracle that always knows the next desired position.

To implement this global route oracle, the Raspberry Pi was connected to the Google Maps API. Using the bicycle's initial position given by the on-board GPS module and a specified destination, the Pi queries the Google Maps API for a route between them. This returns a relatively large data packet. Within this packet is a list of intermediate (latitude, longitude) pairs along the route. These are in some cases too far apart for the local route black box to reasonably handle, however the returned packet also provides a Polyline object. This Polyline is an encoded set of

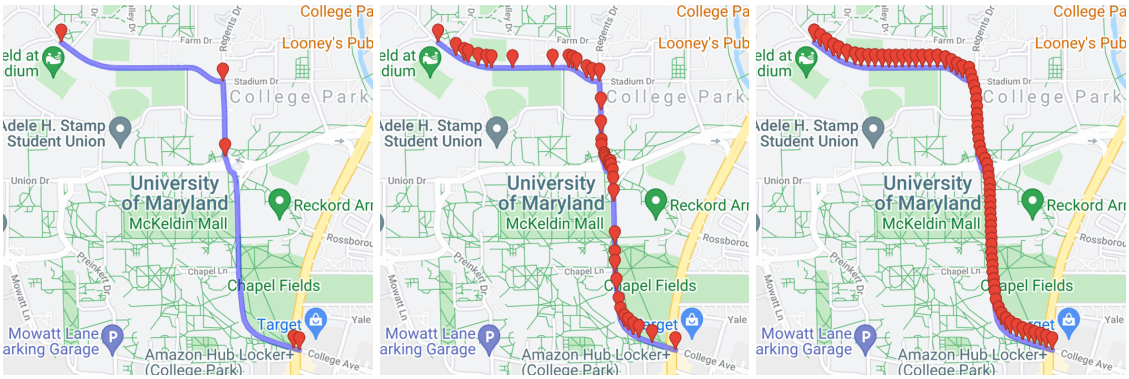


Figure 3.21: The three maps display the same overall route with different amounts and densities of intermediate points. The first on the left shows the intermediate points returned directly from the Directions API endpoint from Google Maps. The middle map displays the points contained in the Polyline. The final map shows the second map augmented to remove points that are too close and with points inserted where points were originally too far apart.

(latitude, longitude) pairs that provides much more frequent intermediate points. Using the points in the Polyline also has issues. Some points in the Polyline are too close together, often within 5 meters of each other, while other points will still be too far. To solve this problem, points that are too close can be removed, and points can be inserted into the gaps simply by linear interpolation. This, finally, results in consistent and useful intermediate positions. These three tiers of intermediate points are visualized in 3.21.

3.11.3 LiDAR Frame Synchronization

The on-board LiDAR is constantly collecting data on the environment. Because the points collected by the LiDAR are always relative to the position of the bicycle, it is crucial that stale points not be processed by the rest of the navigation loop. This is necessary because the navigation loop is designed with the assumption that the points' position are stationary relative to the position of the bicycle. To ensure that only fresh point clouds are fed into the navigation system, a relatively simple algorithm is run. The LiDAR continuously collects frames, each correspond-

ing to a point cloud. Each frame is stored in a local variable and is overwritten every time a new frame is scanned. Once the navigation loop is done processing the LiDAR data, it sends a ROS message to the *LiDAR Driver* node that then queries the LiDAR SDK for its latest frame via the socket that connects them. Upon receiving the query, the LiDAR outputs its latest stored frame to the socket.

3.11.4 Obstacle Detection

The obstacle detection portion of the Navigation system consists of a pipeline that takes raw LiDAR data and processes that data into a series of objects in the bicycle's environment. This section will describe the processes that take place within this pipeline.

3.11.4.1 Point Cloud Interpretation

The LiDAR SDK sends cloud as in a binary format. The exact format for this binary sequence is defined in the LVX file specification provided by Livox. Because the information is encoded this way, an intermediate parsing process is required before the collected data can be analyzed. This process can be found in the *Navigation Core* node. Because the parsing follows the LVX file specification, if the LiDAR model is changed, this process may need to be modified.

3.11.4.2 Defining the Output

For local navigation, a collection of positions and dimensions of surrounding objects must be captured to enable avoidance of objects. The first concept investigated was the You Only Look Once (YOLO) algorithm. This concept had some issues regarding the needs of the Autocycle.

The object detection of YOLO is very accurate but in a non-standard environment where the bike needs to dodge a rusty, bent nail sticking out of the ground, it does not provide the necessary information to allow the bike to avoid the obstacle. Furthermore, given the bike's self-stability, precise and accurate features and distances must be found to enable the bike to avoid objects while maintaining self-stability. As such, we decided to develop an algorithm to extract these features from a LiDAR point cloud. The algorithm developed for object detection would be made in such a manner to enable additional inputs such as labels from an object classification algorithm. Labels from object classification algorithms would allow for special behaviors to be implemented regarding moving versus static objects and for following traffic laws.

3.11.4.3 Defining the Input

The input to our algorithm was a "frame" from our LiDAR. A frame from the LiDAR is a point cloud and consists of (x,y,z) values, reflectivity, roll, pitch, etc. This LiDAR "frame" is translated and its information reduced to only (x,y,z), as that is what we determined was valuable to our algorithm. Certain features such as reflectivity could have proven to be valuable information if we had the time to conduct more research into understanding them. For example, reflectivity could be used to detect certain highly reflective objects such as traffic signs.

Following the translation of a "frame" into a list of (x,y,z) values where we defined x as the horizontal axis, y as the vertical axis, and z as the axis perpendicular to the LiDAR lens, we begin the filtering process.

Given the constraints of our processing power, we filter out data that seems less important or information that could provide greater inaccuracies. The first step for us is to define a minimum

y value to address errors within our LiDAR which produced points that were underground. This filtering could cause issues such as with detection of objects below the crest of a hill which is addressed in the performance subsection of pitfalls. Following this, we filtered max x, y, and z values. A bicycle only needs a certain amount of vertical clearance. We also found as x, y, and z values increased in distance from the LiDAR, point density decreased which added more inaccuracy. To mitigate this we set maximum x, y, and z values of consideration to enable better precision of results while still having a sufficient viewing frame to detect objects.

3.11.4.4 Principles of the Algorithm

The object detection algorithm is based on the computer vision concept of image gradients. The concept of image gradients usually pertains to the numerical differences of adjacent pixels in a grey scale image. In our implementation the pixel values are distances extracted from the LiDAR point cloud. To used a point cloud as an input, a 2D array is created. Each cell within the array corresponds to an area of x and y coordinates. The list of points is iterated through placing points in the cell corresponding to their x and y values. Each cell uses minimum pooling in terms of z value to have the smallest z value assigned to them. Using this data structure, the algorithm searches through columns to find consecutive cells with z values within the parameters set. The current parameters ensure that given two vertically adjacent cells, the cells are classified as the same object if the angle of the vector between the two points is less than 45 degrees away from the vertical axis. In other words, the difference in distances between vertical cells must be less or equal to that of the cell size. If a certain amount of cells in a row are consecutively within the constraints they will be flagged as a key point for that x value of that column. A key point

can also be flagged if there is a large jump in z value towards the LiDAR. This continues to the end of the column and the minimum z of a key point is taken as the z value for the column. The list of key points are horizontally iterated through testing adjacent z values and if they are within the tolerance of each other, they are linked together to become an object, an object is defined by the two end points of a line segment. For each new object, the list of previous objects are iterated through, checking to see if the new object intersects or is inside of the bounding box of the previously recorded objects. Each new object is checked against the bounding box of previous objects ensuring the new segment is not inside the bounding box nor does it intersect the bounding box. This is done to prevent accumulation of the same object though it is not the best practice as indicated in the following limitations and potential solutions subsection. Groups of points are created such that the end points of every new object found in object detection that does not intersect an old object are in a group. Objects are grouped together such that for every object in the group, the group contains one other object that is less than a threshold distance from said object. Every end point of the new objects is added to a graph as a vertex. If points are within the same group, an edge is created between them. Following this we find an object group through finding the different connected components of the graph. The convex hull of each connected components is found to create a convex object using a standard convex hull algorithm.

3.11.4.5 Limitations and Potential Solutions

This algorithm has some different limitations currently that have not been addressed as they do not impact basic functionality but were seen as a means of improvement to the current algorithm.

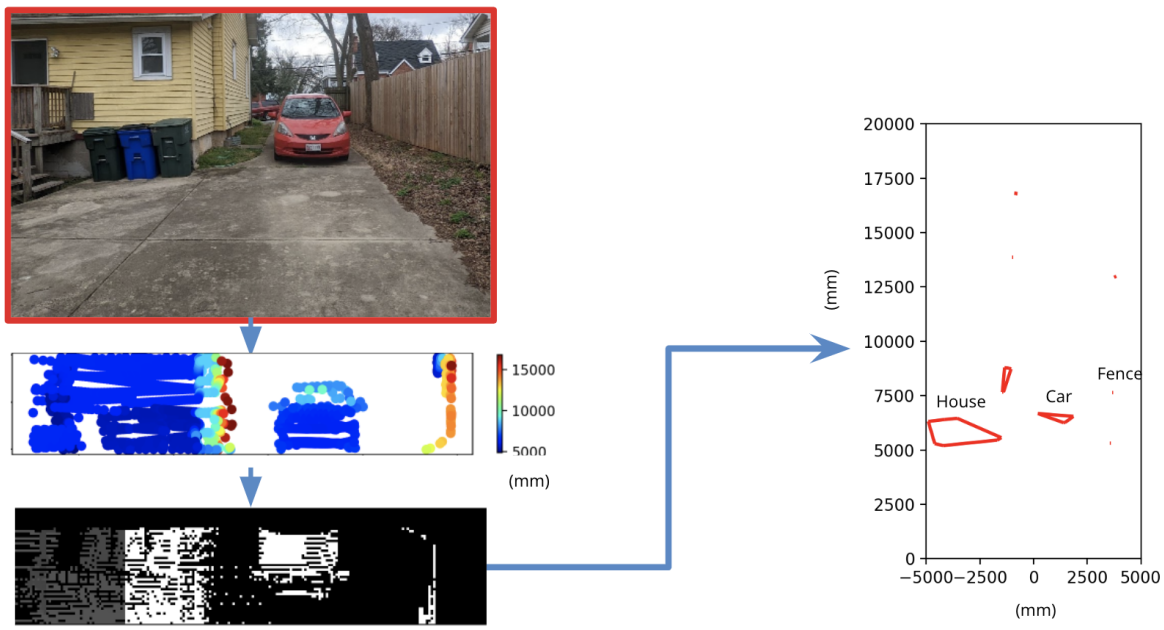


Figure 3.22: The object detection pipeline. First image is the view from the LiDAR. Below is the LiDAR point cloud where the color mapping is defined by the z axis. Below this image is the filtered 2D rendering of the point cloud with the where the black indicates distance 0 and white indicates the maximum distance. The right image is the results of using the algorithm described above.

One limitation of this algorithm is that a single column cannot see multiple objects within it. Currently, the algorithm finds the minimum key points of a column instead of all key points of a column. This was done since a method of processing all these key points was not originally obvious. This meaning that if a person is sitting and someone is standing behind them, it would only see the person sitting closest. To fix the this, the algorithm can be modified to take a list of key points which then iterate over the grouping and convex hull piece instead of using the horizontal adjacency checks. In solving this issue, the algorithm would also could have one small change to have previous objects be combined with currently found objects before being run through the convex hull, resulting in objects becoming more detailed as more frames are processed. This change would not only enable multiple key points in a column but would replace the issues regarding new and old object overlapping. Instead of objects being more definite, future

LiDAR frames would update the dimensions of objects. This means that a bus could at first only be seen as a line segment, but as the bike attempts to avoid it, new LiDAR data would change the object dimensions to show the bus had depth, too, and the path would then update. Using both new and previous objects in the graph and convex hull would produce a better rendering of the environment for path planning.

Another limitation to the algorithm is that based on the input sometimes objects are missed as the point density, the number of points per cell of the algorithm, is not large enough for the algorithm given the desired parameters. This limitation could be addressed through a form of Gaussian smoothing or other types of kernels for convolution. This could impact would be the accuracy of the object location making it less blurry. The major reason this was not used was because convolution does not necessarily fix point density if there are issues regarding cells missing points entirely. To fix the issue regarding missing data, interpolation would need to occur, instead of smoothing, to fix the points per cell issue. Increasing the size of cells could work to help with either issue, but it would reduce the ability to detect smaller objects that could impact the stability of the bicycle such as a random small object on the ground. Overall, it would be beneficial to have more points and a better distribution of points in the area that we sample.

One area that we planned to improve was distinguishing between moving and static items as the LiDAR does not provide enough frames or data to have one frame of data indicate the state of the bike surroundings. Currently the LiDAR can only provide 5 frames per second which would not be sufficient for the bike to react and path plan well when the environment has many non-static objects. Thus, our algorithm keeps previously found items in the environment. By adding tags from image classification, the algorithm could determine if an object should be kept or removed, and if an object is static or non-static. Currently a moving object could ”paint” a wall

of virtual objects across the bicycle's FOV causing the path planning algorithm to break down. Further actions could be taken to predict the non-static object's locations instead of re-sampling the data with the each LiDAR frame.

3.11.5 Bike Position Coordinate Conversion

For local route planning to operate correctly, the position of the bike needs to be known. As it stands the navigation loop continuously receives (latitude, longitude) positions from the GPS module. Though we receive position in latitude/longitude, the points in the point clouds returned from the LiDAR are centered about the LiDAR module and are in meters. This means that obstacles detected are represented by their position in meters. It is necessary for the bicycle position to be in meters as well to be able to accurately generate paths around obstacles. To account for this, the (latitude, longitude) pairs must be projected onto a 2d plane using meters as the unit. This projection is performed by taking the starting position of the bicycle to be the origin. All subsequent positions are calculated by taking the GPS position and calculating the distance between the bike's current (latitude, longitude) position and the previous one and calculating a point at that distance away from the previous position in the direction of the current heading of the bicycle. Equation 3.2 shows how to calculate the distance between two (latitude, longitude) pairs. Equation 3.3 shows how to bike position is updated given the previous and current (latitude, longitude) pairs. For both of these equations the angular units are assumed to be radians.

$$\Delta \text{lat} = \text{lat}_2 - \text{lat}_1$$

$$\Delta\text{lng} = \text{lng}_2 - \text{lng}_1$$

$$\alpha = \sin\left(\frac{\Delta\text{lat}}{2}\right) * \sin\left(\frac{\Delta\text{lat}}{2}\right) + \cos\left(\text{lat}_1\right) * \cos\left(\text{lat}_2\right) * \sin\left(\frac{\Delta\text{lng}}{2}\right)^2$$

$$c = 2 * \arctan(\frac{\sqrt{a}}{\sqrt{1-a}})$$

$$\text{distance} = r * c \tag{3.2}$$

where r is the radius of the Earth in meters

$$d = \text{distance}(\text{previous\_latlng}, \text{new\_latlng})$$

$$\text{x\_diff} = d * \cos\left(\text{heading}\right)$$

$$\text{y\_diff} = d * \sin\left(\text{heading}\right)$$

$$\text{bike\_position} = (\text{bike\_position}.x + \text{x\_diff}, \text{bike\_position}.y + \text{y\_diff}) \tag{3.3}$$

3.11.6 Local Route Planning

For the bicycle to follow the path generated by the global route planner, the bicycle must be able to get between the intermediate points specified by the global route planner. Given the bicycle's current position and a goal destination served by the global route planner, the bicycle must be able to plan a path to the destination avoiding any obstacles in its way. The solution uses an A* traversal scheme based on the work by [46] described in 2.

The area around the bicycle is defined as a grid consisting of equidistant nodes. this A* traversal scheme, the goal is to get from some start node at or near the bike position and an end

node that approximates the desired destination, however nodes that would overlap with detected obstacles would be pruned prior to traversal. Crucially, the position of the bike and the position of objects do not necessarily lie nicely on this graph. As such, methods were implemented that approximate any coordinate position to the node coordinate system. This pruning occurs by blacklisting nodes every time a new object is detected. Below is a high level overview of the process used to generated blacklisted nodes from a new object.

1. Choose some safety parameter $k \in \mathbb{Z}$
2. Assume there already exists a set of blacklisted nodes B
3. Consider the new object with end points (x_1, y_1) and (x_2, y_2)
4. Consider (x_1, y_1) the 'current node'
5. Add the node that best approximates the current position and all nodes within radius k to the set of blacklisted nodes.
6. Consider the line between (x_1, y_1) and (x_2, y_2), and choose a point along the line that is a length of a node away from the previous considered point.
7. repeat 5-6 until a the new point no longer lies on the line. Repeat 5 for (x_2, y_2) if it has not already been considered.

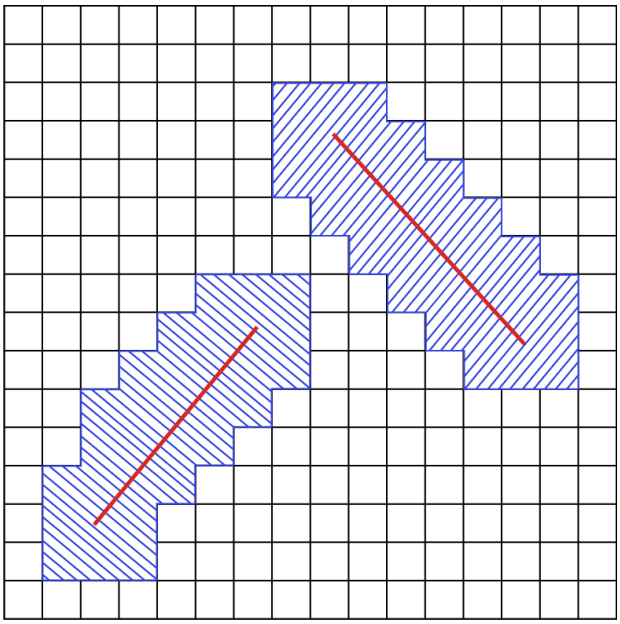


Figure 3.23: An example graph after nodes near obstacles have been pruned for graph traversal. In this graph, each black line intersection describes a node. The red lines represent obstacles and the dashed blue areas represent areas where nodes have been pruned.

Before describing the A* implementation in any depth, the neighbor node generation for the graph should first be described. Unlike a typical graph, this implementation procedurally determines the neighbors a given node has. For any given node, the possible subsequent nodes are determined by the position of the previous node. Throughout the development of the Autocycle the way these neighbors are generated has changed. The first of these ways which we refer to as 'grid-locked neighbors' is shown visually in 3.24.

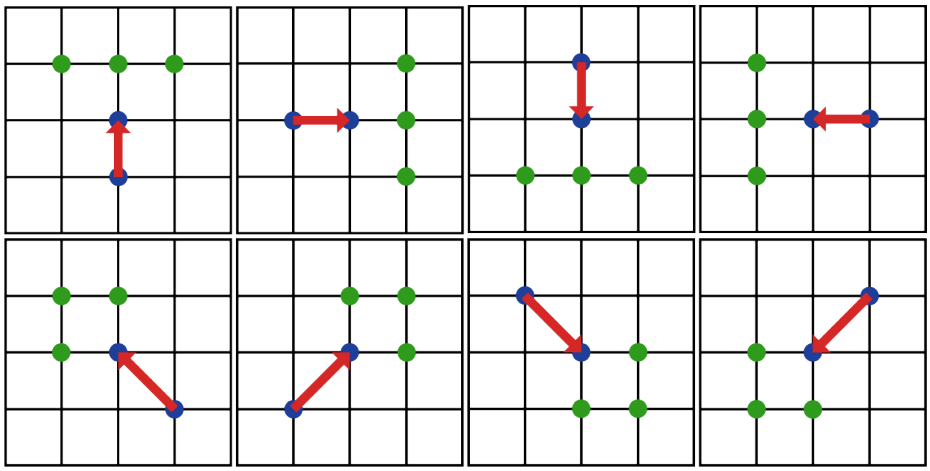


Figure 3.24: Shows the allowable neighboring nodes in the A* traversal for each possible condition. The blue circles represent the current and previous node with the previous node at the base of the arrow and the current node at the point. The green nodes represent the possible options for the following node.

The neighbors are selected in this way to both reduce the size of the traversal and to prevent 90 degree turns. Preventing 90 degree turns results in paths that are smoother after interpolation and are more feasible to achieve.

An issue with this node traversal method is that paths can be generated where the turning radius at some point along the path is beyond what is physically possible given the physical properties of the bicycle. To account for this, a separate method of neighbor node generation is employed which we refer to as 'kinematic neighbors'. Again, a visual for this node generation is shown in Figure 3.25.

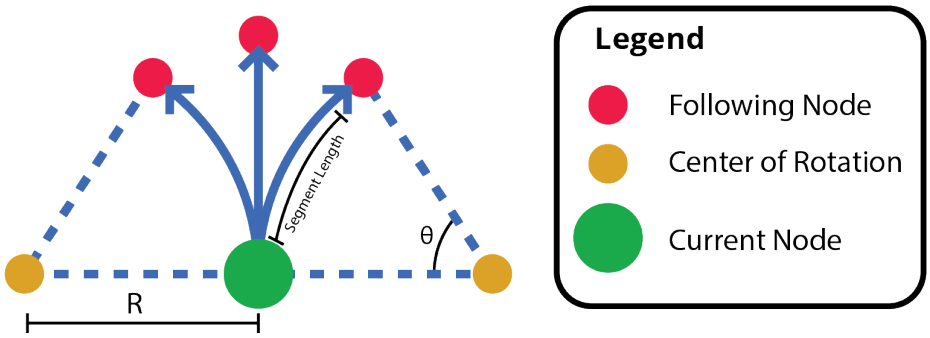


Figure 3.25: Shows the neighbor generation for nodes taking the kinematics and physical properties of the bicycle into account. R represents the turning radius. θ represents the amount of a semi-circle to generate such that the length of the arrow is $segment\_length$.

In this neighbor generation scheme, R is calculated by solving for R in Equation 3.4 instead of δ. The maximum steering angle and the current velocity is substituted for δ and v respectively. The distance the neighbor points are from the starting point is dictated by the parameter $segment\_length$. This parameter is set somewhat arbitrarily with lower values increasing resolution at the cost of computation. An interesting side effect of using this method is that nodes are no longer constrained to be on the grid. Using this method, the grid is only used to model which parts of the environment are blocked off by obstacles. The neighboring nodes $p1, p2, p3$ are calculated using the equations below.

$$[H]R = \frac{L + (\frac{W_f}{C_{\alpha f}} - \frac{W_r}{C_{\alpha r}}) \cdot \frac{v^2}{g}}{\delta}$$

$$cr1 = (R * \cos(H + \frac{\pi}{2}) + n_x, R * \sin(H + \frac{\pi}{2}) + n_y)$$

$$cr2 = (R * \cos(H - \frac{\pi}{2}) + n_x, R * \sin(H + \frac{\pi}{2}) - n_y)$$

$$ang_1 = \frac{segment\_length}{R} + (H - \frac{\pi}{2})$$

$$ang_2 = -\frac{segment\_length}{R} + (H + \frac{\pi}{2})$$

$$p1 = (R * \cos(ang_1) + cr1_x, R * \sin(ang_1) + cr1_y)$$

$$p2 = (segment\_length * \cos(H) + n_x, segment\_length * \sin(H) + n_y)$$

$$p3 = (R * \cos(ang_2) + cr1_x, R * \sin(ang_2) + cr1_y)$$

Where in addition to the variables already defined in Equation 3.4,

- H = The heading of the bicycle at the position neighbors are being generated for
- $cr1$, $cr2$ = The centers of rotation as shown in Figure 3.25
- ang_1, ang_2 = Angles at which final points $p1$ and $p3$ should be placed from $cr1$ and $cr2$ respectively
- n_x, n_y = The position of the position neighbors are being generated for

Note that to calculate the neighbors for some position on the grid, the heading of the bicycle when the bicycle is at that point is required. Obviously, this will differ significantly from the initial heading. To facilitate this, headings h_1, h_2, h_3 are associated with the neighbors $p1, p2, p3$.

$$h1 = H + \frac{segment\_length}{R}$$

$$h2 = H$$

$$h3 = H - \frac{segment\_length}{R}$$

The A* traversal will not search further than 50 nodes away from the bicycle to try to find a path to the destination node. Another feature this modified A* algorithm provides is that if

a path to the desired position cannot be found, a path to the closest node to the end position is returned instead. This is done in the consideration that the 50 node restriction is the limiting factor rather than an actual lack of viable path. If the closest node is within 5 meters of the bike, then it is determined that there is not actual path and the bike is told to stop immediately. The A* heuristic used is simply a linear combination of the length of the path and the distance to the destination point. The weights of those variables are 1.0 and 1.2 respectively. Below is a high level look at how the A* search works without bloating details on how neighbors are computationally computed.

The Autocycle uses a separate method of path generation in the event there are no obstacles to avoid. The first thing the pipeline does when a new path is needed is to draw a straight line from the bicycle to the end position. If there are no obstacles intersecting this line, this line is used as the path. This has the added benefit that the end position does not have to be approximated by a node. Another important point is that it is not beneficial to generate a new path if it is not needed. Regenerating paths constantly can result in unpredictable behaviour from the bicycle. As such, a path is only generated in one of the following situations.

1. The bicycle has strayed too far from the existing path
2. A new obstacle has been detected that blocks the current path
3. A direct path to the end position has become available
4. The bike has arrived to the end position

In the case that the bike arrives to the end position, the *navigation core* node queries the *global route oracle* node for the next end position. If none is returned, it is because the bicycle

Algorithm 1 A* Graph Traversal

$q \leftarrow PriorityQueue$ ▷ Popping queue returns item with lowest value
$q \leftarrow (0,\ start\_node,\ [bike\_pos, start\_node])$ ▷ queue item values: score, node, and path
$best\_path \leftarrow [start\_node]$
$best\_dist \leftarrow euclidean\_distance(start\_node, end\_node)$
$visited \leftarrow Set$
while $!q.empty$ **do**
    $cur\_val,\ cur\_node,\ cur\_path \leftarrow q.pop$
    $neighbors \leftarrow get\_neighbors(cur\_node)$
    if $cur\_node == end\_node$ **then** ▷ end node found!
        $return\ cur\_path$
    end if

    if $cur\_node \in visited\_nodes$ **then**
        $continue$
    end if

    for $neigh \in neighbors$ **do**
        if $neigh \in blacklisted\ ||\ neigh \in visited\ ||\ neigh \in q$ **then**
            $continue$ ▷ neighbor either blacklisted, visited, or already to be visited
        end if

        if $|neigh.x - start\_node.x| > 50\ ||\ |neigh.y - start\_node.y| > 50$ **then**
            $continue$ ▷ neighbor outside of bounds
        end if
        if $euclidean\_distance(neigh, end\_node) < best\_dist$ **then**
            $best\_dist \leftarrow euclidean\_distance(neigh, end\_node)$
            $best\_path + neigh$
        end if

        $line\_len \leftarrow euclidean\_distance(neigh, cur\_node)$
        $end\_dist \leftarrow euclidean\_distance(neigh, end\_node)$
        $neigh\_value \leftarrow cur\_val + 1.0 \cdot line\_len + 1.2 \cdot end\_dist$ ▷ value for current neighbor
        $q \leftarrow (neigh\_value, neigh, cur\_path + neigh$
    end for
end while

if $euclidean\_distance(best\_path.end, bike\_pos) < 5$ **then** ▷ No viable path exists
    $stop\_bike$
end if

$return\ best\_path$

has arrived to the final position and the bicycle stops.

The final point of interest for local route planning is the discussion of coordinate systems. There have been two major iterations throughout the development of the bicycle: a bicycle centered coordinate and a global coordinate system.

The bicycle centered coordinate system always places the bike at the origin with the graph nodes in line with the heading of the bicycle. This is advantageous for the A* traversal algorithm since it means that the bicycle is always laying directly on a node at (0, 0). It also allows for direct interpolation of the path without having to transform the path in any way ahead of time. This method has many drawbacks. Because the bicycle never moves, all obstacles have to be constantly updated and moved depending on how the bike has moved since the last update. Getting this update system to work proved challenging and was riddled with errors. Up until there was a GPS module on the bicycle, this was thought to be the most viable option since a global coordinate system would require accurate knowledge of where the bicycle was in Earth-centered space.

The second and current iteration of the coordinate system in the global coordinate system. Here the bicycle is not locked at any point. Obstacles on the other hand, once placed on the graph, never move. This system has a number of advantages over the bicycle centered coordinate system in that the only thing that needs to be updated is the bicycle's position. There are still a number of drawbacks, though they pale in comparison to the issues faced in the bicycle centered system. One of these drawbacks is that the obstacle detection pipeline is still detecting objects in a bicycle centered coordinate system since that's how the LiDAR points are collected. Consequently, every time a new obstacle is detected it has to be rotated and translated according to the position and heading of the bicycle. This is an operation that has to occur only once for each obstacle. The

second drawback is that the bicycle is no longer guaranteed to be on any given node. To address this second issue, the closest node is used as the start node. The algorithm is biased to choose a start node in the general heading of the bicycle to make the path more viable. A line is then drawn from the bicycle position to the chosen start node and is prepended to the path. An example path using grid-locked neighbors can be found in Figure 3.26 or kinematic neighbors in Figure 3.27.

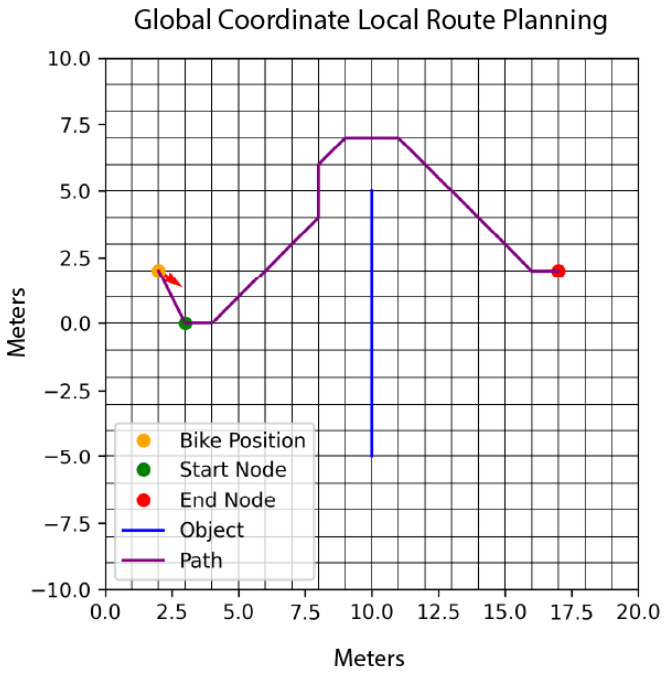


Figure 3.26: A path planning instance where grid-locked neighbors are chosen rather than kinematic neighbors.

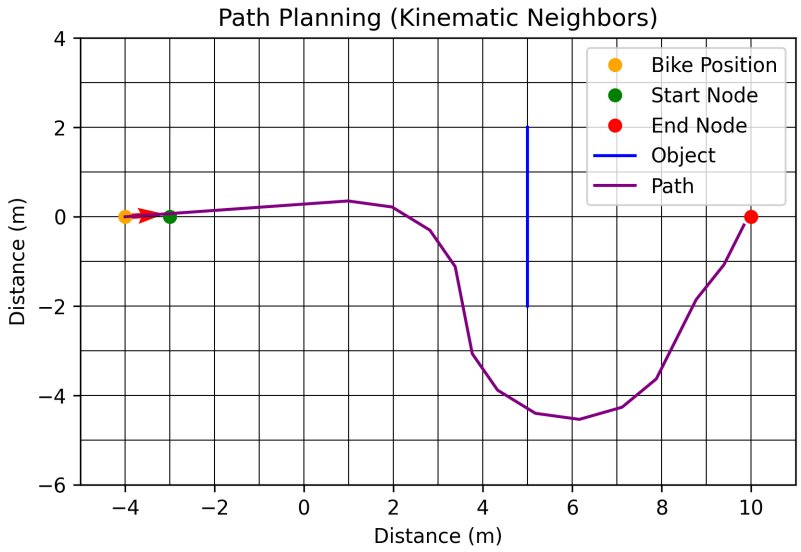


Figure 3.27: A path planning instance where kinematic neighbors are chosen rather than grid-locked neighbors.

3.11.7 B-Spline Path Interpolation

In order for the bicycle to follow a generated path, the path needs to be converted into a series of commands to be sent to the Arduino and lower level controller. Before this is done, the path needs to be smoothed. Having a path of straight line segments makes it more difficult to perform any kind of derivative analysis that would be needed to generate these instructions. To perform this smoothing the *scipy* library is used. The library is used to generate a B-Spline representation of an N-D curve. The path generated can be fed into this library to produce a smooth parameterized curve that is differentiable. Figure 3.28 shows the result of passing 3.26 through the library. Derivatives along the parameterized path are estimated using nearby points rather than symbolically to save on computation.

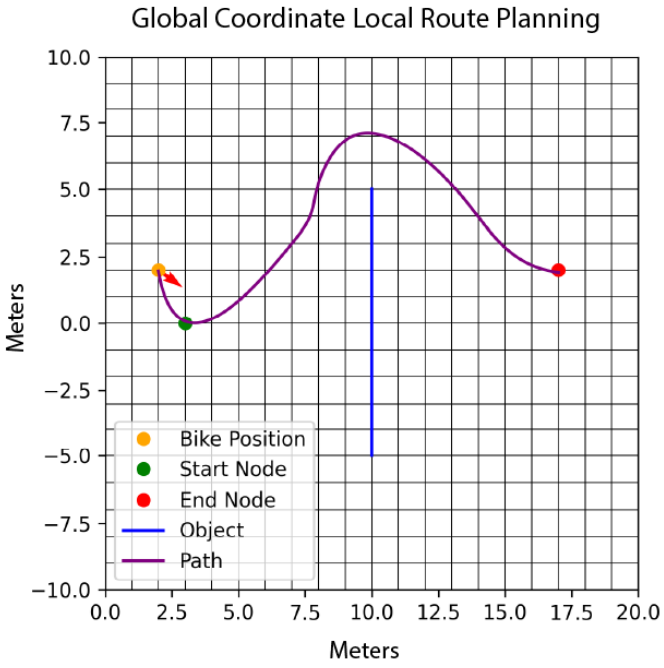


Figure 3.28: Shows the result of passing 3.26 through a B-Spline interpolation function

3.11.8 Steering Angle Calculation

Once a path has been interpolated, that interpolation is converted into a series of desired steering angles over the course of the path. The parameterized curve can be evaluated from start to end using values between 0 and 1 inclusive. Equation 3.4 shows how to generate a steering angle for a given point.

$$R = \frac{(x'^2 + y'^2)^{\frac{3}{2}}}{(x'y'') - (y'x'')}$$
$$\delta = 57.3\frac{L}{R} + \left(\frac{W_f}{C_{\alpha f}} - \frac{W_r}{C_{\alpha r}}\right) \cdot \frac{V^2}{gR} \tag{3.4}$$

Where

- x, y = Values returned from evaluating the parametric curve at some value $k \in \mathbb{R}\ s.t.\ 0 \leq k \leq 1$
- x', y', x'', y'' = First and second order derivatives of the parametric at the same point k
- R = Turn radius at point k (m)
- L = Wheelbase (m)
- V = forward (m/sec)
- g = Gravitational acceleration (m/sec^2)
- W_f = Load on front axle (kg)
- W_r = Load on rear axle (kg)
- $C_{\alpha f}$ = Cornering stiffness for the front tire (kg_y/rad)
- $C_{\alpha r}$ = Cornering stiffness for the back tire (kg_y/rad)
- δ = Steering angle (rad)

As the bike is operating, it keeps track of where it is on a given path and sends the appropriate steering angle to the bicycle. Figure 3.29 shows the steering angles along the same path shown in 3.26 and 3.28.

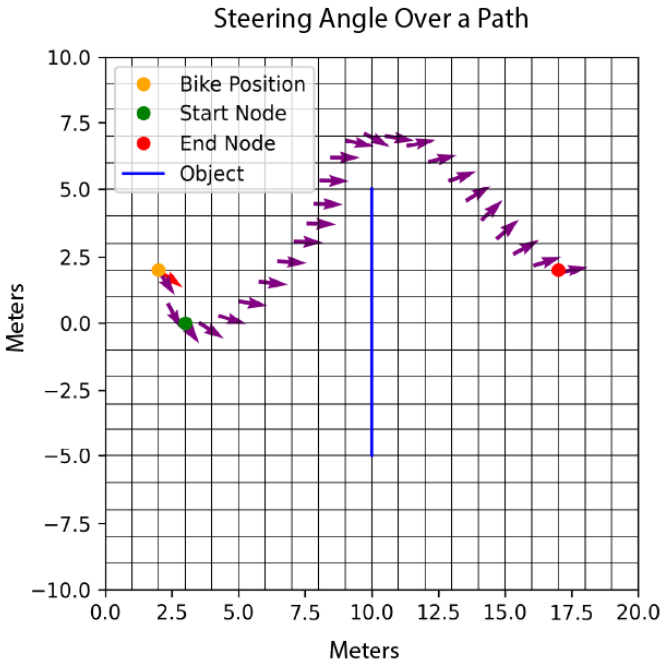


Figure 3.29: Displays steering angles over the course of the path. A zero steering angle is oriented towards positive x-axis; That is to say the steering angles are not oriented relative to the slope of the curve.

3.11.9 Serial Communication

In order for any of the operations for navigation to work properly, the Raspberry Pi must be able to receive telemetry data from on-board sensors and send instructions to the motor controllers. All of these functions are managed by the Arduino Due. Communication between the Raspberry Pi and the Due is performed over a serial connection. Instructions are sent using labeled arguments and telemetry data is received tab separated on new lines.

3.12 Radio Communication

During a portion of the test campaign, radio communication was used to transmit telemetry and commands. This was accomplished over the 2.4 GHz ISM band using an NRF24L01P-R7

radio and a simple wire protocol. However, due to congestion on the band and possible electrical faults, this method proved to be unreliable. As alternatives were available (802.11 WiFi via the Raspberry Pi, wired communication + disconnect), this method was abandoned in the winter of 2022.

3.13 Weathering of Components

In order for an autonomous bicycle to be a practical and sustainable business solution for bike-sharing systems, it is favorable to minimize maintenance requirements and frequency as much as possible. ”Wear-and-tear” effects, such as rust buildup, can compromise the bicycle’s appearance and appeal to potential users, and compromise the strength and structural integrity of materials, making the bicycle’s mechanical systems more vulnerable to failure. Because the bicycle operates outdoors, exposure to elements must be considered and mitigated through anti-weathering strategies. One such strategy is selecting materials that are naturally weather-resistant, however availability of materials and costs can limit the practicality of this strategy when manufacturing components and systems in-house. Many of the bicycle’s preliminary parts and systems were fabricated with steel and had begun to rust after 1-2 years of use. Rather than re-fabricating these components, a time-consuming and expensive process, it was decided to treat the components for rust removal and weather resistance.

To ”de-rust” the affected parts, most notably the steel components of the ZSS and torque motor belt tensioner, the systems were first disassembled into their individual components. A gel de-rusting solution was then applied to all affected regions of the parts. The parts were then wrapped in a plastic film and left for 24 hours. After 24 hours, the plastic film was removed,

revealing the successful elimination of any visible rust from all components, and the parts were cleaned with ethanol and prepared for bluing treatment to prevent future rust development.

Hot bluing methods involve heating steel parts to temperatures exceeding 275°F and submerging them in oil. The outcome of this process, if done correctly and within the proper temperature range, results in a non-corrosive, protective black oxide coating of Fe_3O_4 surrounding the surface of the steel part. This coating is non-reactive with oxygen and effectively seals the steel beneath from reacting with oxygen and forming corrosive Fe_2O_3, known as rust.

While hot bluing is inexpensive and generally understood as the industry standard for manufacturing, the method has its disadvantages. For one, heating parts using amateur equipment such as a blowtorch and metal tongs, especially when conducted by untrained individuals, can pose safety risks. Small or thin parts are also vulnerable to warping when heated, and geometric limits are constrained by the size of the oil container for submerging. To avoid these concerns, a cold bluing procedure was instead pursued.

Cold bluing achieves the same effect as hot bluing, but does not require any heating. Cold bluing involves topically applying a selenium dioxide solution to a metal part. For this, Birchwood Casey Super Blue was selected due to its affordability, availability, and reputation as a market standard. The cold bluing procedure for each part was conducted as follows:

1. De-grease the part by applying acetone to its surface with a paper towel
2. Use a a brush to apply an even coat of bluing agent to the part's surface
3. Once the entire surface's finish appears black, submerge the part in cold water
4. Dry the part with a paper towel or compressed air and apply oil to its surface

5. Let the part rest for at least 12 hours

The results of de-rusting and bluing the belt tensioner's steel components are shown in Figure 3.30.

(a) Before processing (b) After processing

Figure 3.30: Belt tensioner components before and after de-rusting and bluing.

3.14 Presentation and Wiring Harness

A small aspect of finalizing the chassis was organizing the wiring and making the bicycle accessible for a human rider. Our ultimate vision is to see the Autocycle used in a bike share system, so a prospective rider must have the ability to safely mount and use the Autocycle like a typical electric bicycle without damaging the wiring. In order to achieve this, groups of wires were sheathed together to minimize the number of safety hazards where a rider could get caught. Placing wires in a sheath also grants another layer of protection, and reduces the risk of bad weather or abrasion from collisions damaging the wires that control the Autocycle and its components.

It was also deemed necessary to enclose the central frame triangle, the area inside the down tube, top tube, and seat tube. This was to protect our electrical boards and motor controllers from the elements and prevent the rider from interfering with those components. In order to minimize weight and implement an elegant and visually appealing design, it was decided to use a fabric covering around the triangle. The top tube is left exposed to prevent damage to these components caused by the fabric laying on top of them. The fabric is adhered to the chassis using adhesive hook-and-loop strips that allow easy removal of the fabric when it is needed to access internal components. This design is intended for the prototyping stage only. A final, marketable design will have a more permanent enclosure.

Chapter 4: Validation & Results

4.1 Simulation

To analyze different control schemes without risking damage to the system, a simulation was created. First the simulation had to be validated with existing data. The bicycle model used in the simulator was originally designed with the model parameters from the Kooijman paper [16] (see Sec. 2.1 and Sec. 3.8.3). The simulator estimated a self-stable region from about 4 m/s to 7.9 m/s where the bicycle was able to right itself from slight perturbations while uncontrolled. As Kooijman et all had observed the same thing for the same bicycle parameters, this confirmed the validity of the implementation.

Next, controls systems were implemented to expand the region of stability and improve the response. To differentiate between successful control systems and unsuccessful ones, the following metrics were considered: Overshoot, Settling Time, and required Torque output. Of all the systems tested, Full State Feedback (Pole Placement) proved to be the most efficient and robust across range of velocities.

Once the parameters for the Autocycle were found (see Sec. 3.2) and a dynamic model was created, the parameters were moved into the simulation. The control system was tweaked to work for the Autocycle which laid the foundation for the final control structure on the physical bicycle.

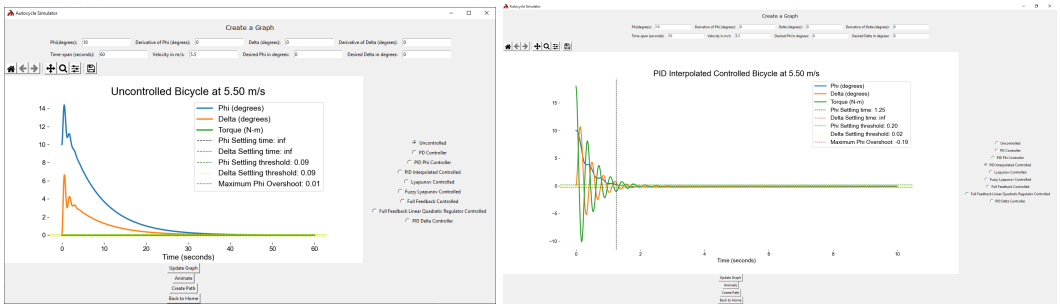


Figure 4.1: The PID controller leads to a much quicker settling time

Alongside the simulated model of the bicycle, an animated visualization was created to better understand how the bicycle was reacting at each point in the control sequence. This was accomplished using python running Panda3D, an open-source 3D engine. Steering and roll data and the derivatives of each were pulled directly into the software, and an on-screen bicycle model would give a visualization of how a bicycle, in the optimal conditions, would look. An example of this can be seen in Figure 4.2.

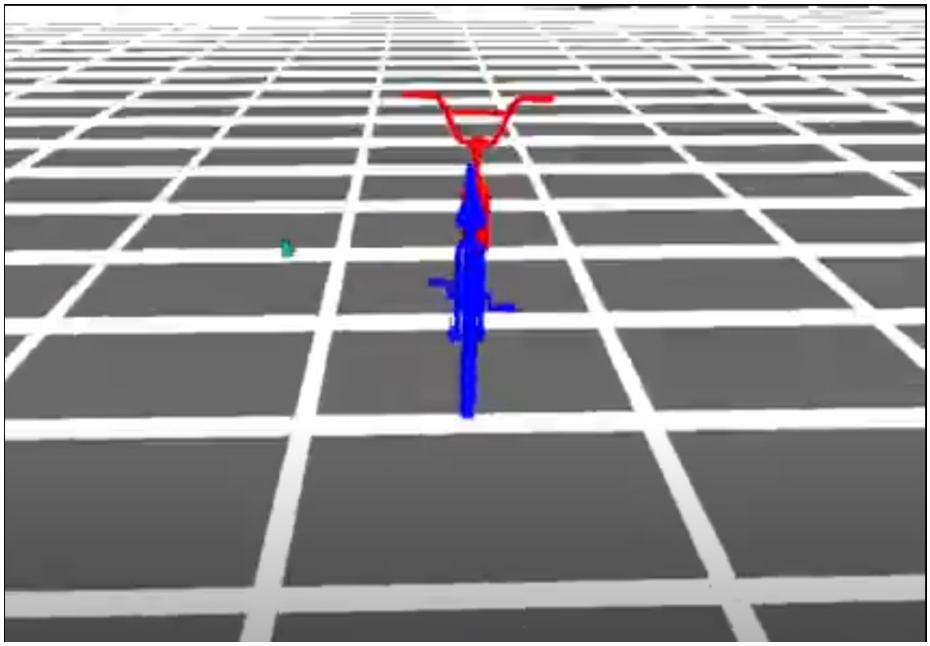


Figure 4.2: Simulation visualization in Panda3D.

4.2 Experiments

The Autocycle testing campaign took place over 2.5 years, beginning in November 2019 once the first components arrived, and concluding in the spring of 2022. Initial testing consisted of design validation of individual off-the-shelf components including various motors, sensing equipment, and parameter identification of the bicycle frame. Next, mounts for these components were created. These included a belt tensioner for the torque motor which controls the bicycle's handlebars, and various electronics across the bicycle.

Initial field testing required various design improvements to improve component resilience. This progressed in parallel with other software and hardware developments, including on-board test data storage and continued off-bike testing of the LiDAR. The team determined that the Autocycle required a deployable training wheel system (the ZSS, Sec. 3.6) to achieve successful stability tests, so this system was designed and manufactured while other design components were tested in the field using standard non-retractable training wheels.

Once the majority of physical components and hardware issues were resolved, testing cadence increased significantly in the Fall of 2021 and allowed rapid iterative testing of control methods for stabilizing the Autocycle as well as navigation and object detection algorithms. The Autocycle first travelled upright under its own control for an extended period of time on December 5th, 2021, just 2 years after the bicycle was acquired and less than a year after the first field test.

In this section, the details of the testing campaign will be delineated, tracing out the process of achieving the upright, self-stable bicycle over the course of the project. We will also discuss our approach to developing a consistent test procedure and the various design iterations and

components additions that had to be addressed to progress.

4.2.1 Component Acquisition, Identification, and Validation

Shortly after the Autocycle was initially designed, all major off-the-shelf parts were ordered and the bicycle frame was received in November 2019 (see Figure 4.3). The bicycle was initially assembled and then disassembled shortly after to conduct center of gravity and identification on each of the four frame components, utilizing the same general approach as Kooijman et. al. described in the literature review [16].

Figure 4.3: The bicycle frame on campus, as initially assembled. As will be seen, the frame was later modified and used as a mounting frame for the Autocycle's electronics.

The center of gravity was initially determined (see Figure 4.4) for each of the four bicycle frame components, but as will be noted later, these values (as well as the later identified moments of inertia) required reevaluation after the bicycle had been more fully outfitted with its suite of sensing and actuating components.

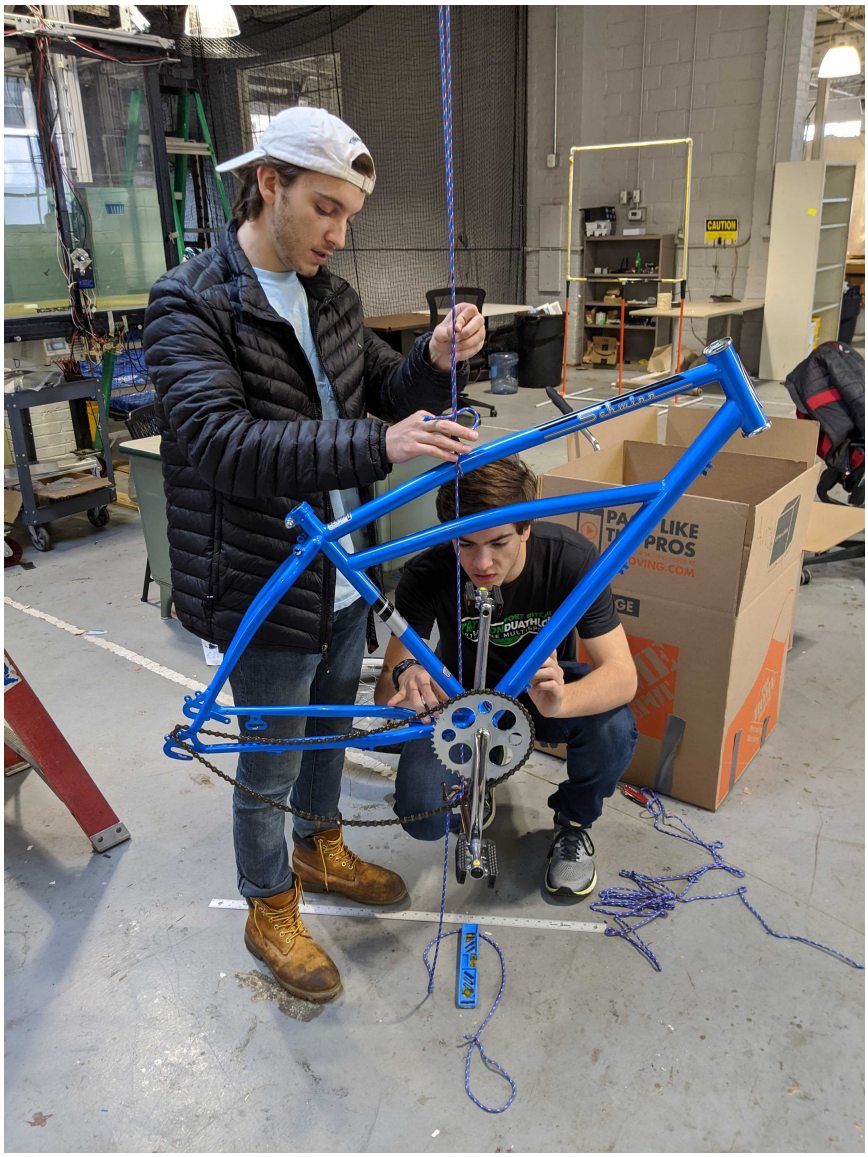


Figure 4.4: One of the four bicycle frame components, the back frame, suspended from a cord to determine its center of gravity. Later tests would use a modified setup with a torsional pendulum so that both center of gravity and moment of inertia could be determined from the test.

Shortly after the start of the next semester, initial power-on testing was conducted with the drive motor, and the bicycle was modified to accommodate motor mounting. The pedals were removed and the central tube which held the pedals was cut down to a width that could accept the drive motor. The motor was successfully powered on and spun up using a temporary battery provided by the team's mentor, Dr. Gomez, which was later replaced with a standard electric

bicycle battery. Testing went on hiatus shortly after these steps due to the onset of the COVID-19 pandemic in February 2020 and the subsequent lack of access to campus facilities and group interactions.

4.2.2 Testing Accommodations during the Pandemic

In order to continue testing in a safe manner, the team pursued several compromises during the course of the pandemic to keep all team members healthy while testing continued. When classes began in the fall of 2020, the testing base of operations for the team was relocated. Originally located in the Cypress building on campus, future testing and in-person work that didn't require specialized machines was conducted in a team member's backyard workshop located not far from campus. This location provided the team with several benefits. Located in a shed, the garage door was left open during testing to maximize ventilation, and all team members wore masks and practiced social distancing. Team gatherings for testing were also limited to only three attendees at a time to minimize risk of exposure. In addition to the health benefits of the relocation, this also provided the team with a consistent storage space for all equipment, which was not possible in the original location. Thus, all future testing (not including field tests) was conducted here.

4.2.3 Further Component Testing and Preparations for Initial Field Testing

Once in-person work resumed in September 2020, progress on several components and electronics systems continued rapidly in preparation for the first motion test. The LIDAR system was tested upon delivery and successfully demonstrated the ability to locate objects within the

expected ranges. Next, the team pursued testing and mount design for the torque motor that controls the front frame of the bicycle. The torque motor was mounted parallel to the stalk of the bicycle using a 3D printed mount, a timing pulley fitted to the shaft of the torque motor as well as the handlebar shaft, and a timing belt connecting the two. The belt requires tension to operate, which resulted in several design iterations using 3D printed material to create a tensioner, which can be seen in Figure 4.5 and Figure 4.6. Once the mounts were printed, the key ways and keyslots machined on each side of the belt mechanism, and the tensioner completed and tested, the torque motor was ready for integration. Shortly after, the torque motor was tested with the bicycle on the ground, and it demonstrated the ability to turn the front frame of the bicycle in both directions.

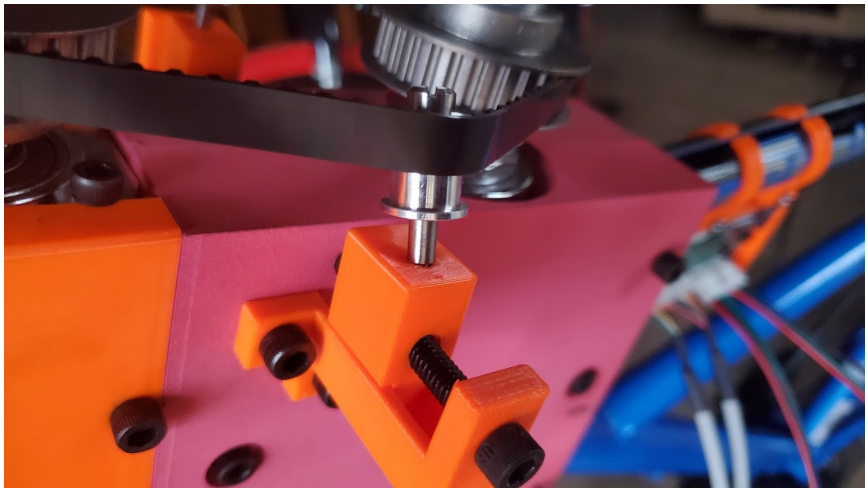


Figure 4.5: The initial design of the torque motor belt tensioner failed due to excessive cantilevered stress on the shoulder bolt supporting the idler pulleys.

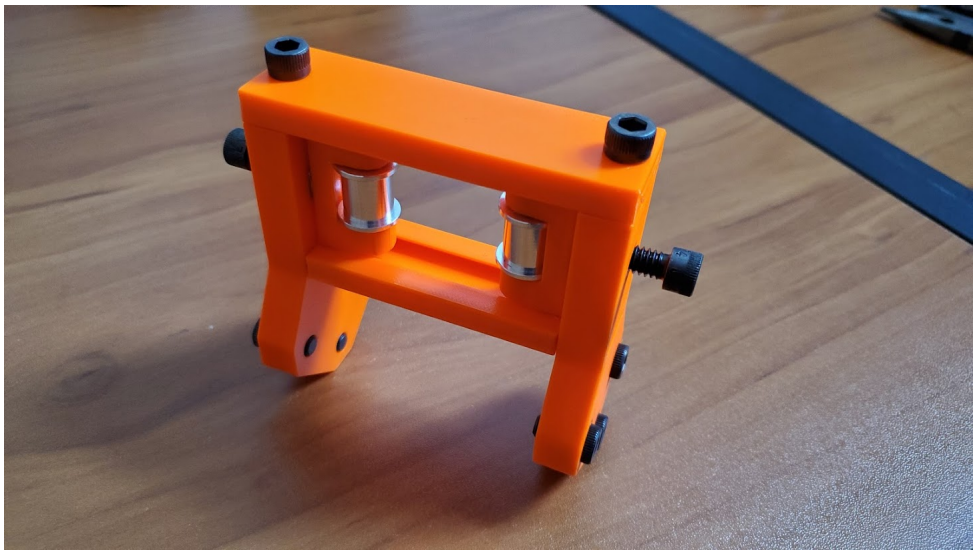


Figure 4.6: In the revised design, rollers were connected with a top bar that removed the cantilevered loading setup which previously lead to failure. This design functioned on the Autocycle until a new material was needed later on in testing.

In order to simplify the electronic mounting process, a bicycle stand was constructed for the Autocycle to rest on during integration. Then, all of the major electronic components were mounted onto the Autocycle, including the Arduino, torque motor controller, and power distribution board. Additionally, the geared hub on the back wheel was removed and replaced with a rigid coupler, which was initially 3D printed.

Figure 4.7: Pictured is the configuration of the Autocycle during initial field testing. It is resting on the test stand constructed by the team.

4.2.4 Initial Field Testing

The first field test of the Autocycle took place on December 11th, 2020. Before the test could occur, scouting took place around the campus area to find a suitable location for testing. The location had to be level, paved, and free of obstacles to maximize ideal conditions for Autocycle testing. The location of the first test was at the College Park dog park parking lot, a location that, during the winter, was generally free of vehicles on weekends when testing occurred.

The first several field tests had the goal of demonstrating the bicycle moving under its own power, with a stretch goal of demonstrating stable upright motion. The bicycle was tethered to a laptop which sent commands, and the experimenters would run alongside the bicycle during motion. In the very first test, motion was not achieved at all. This was determined to be a result of the printed coupler between the back wheel and bicycle chain shearing. The coupler was reprinted with stronger material (TPU), and testing resumed 4 days later on the 15th of December,

2020. This time, the bicycle did achieve motion after receiving a command from the laptop, but quickly fell over and damaged the belt tensioner as it hit the ground, seen in Figure 4.8.

Figure 4.8: Bolt tearout as a result of the Autocycle falling over during initial field tests. This test demonstrated a need for a tensioner robust to impact loading.

Initial testing showed several improvements that would need to be incorporated into the Autocycle design and test procedure before continuing. The first was a need for remote or detachable command of the bicycle. The team determined that a radio module on the Arduino

could serve this purpose, and a suitable option was ordered shortly after. There was also the need for the Autocycle to reach a certain velocity before the testing of the active control scheme could begin. This was the seed for the design and implementation of the Zero Speed Stability system (ZSS), a set of training wheels that can retract and deploy as needed, while the bicycle is in motion. Finally, the necessity of impact resistance for many components of the Autocycle was demonstrated. Although potentially not as important during operation of an Autocycle, the testing phase would see many more falls and the hardware had to be designed to compensate for this as much as possible.

4.2.5 Revised Moment of Inertia Testing

While work continued on upgrading hardware for the Autocycle, moment of inertia measurement was redone for each of the bicycle frame components, now with the relevant hardware also attached to gather more accurate information. A test stand was designed to grip each frame section with 4 screws, which hung suspended from a thin metal rod which was used as a torsional pendulum. As described in the literature review, these tests were conducted and filmed to measure the period of the pendulum and derive the moments of inertia in each axis for the front and back frame (seen in Figure 4.9).

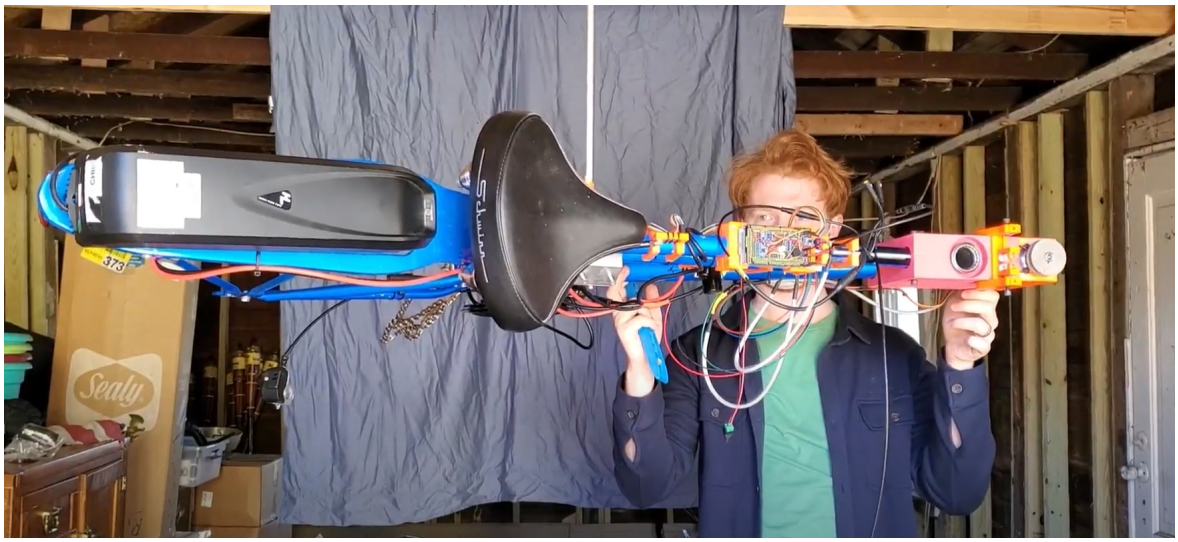


Figure 4.9: Moment of inertia testing on the back frame of the Autocycle. A perturbation is applied in the axis of rotation of the hanging metal rod, and the torsional period of rotation is measured using film playback to determine the moment of inertia around this axis.

4.2.6 ZSS Design Iteration, Construction, and Testing

After moment of inertia identification was concluded, another field test demonstrated that the bicycle could not control itself reliably due to the use of static training wheels. Although we assessed the necessity of support wheels below a certain velocity, once the Autocycle attempts to control its upright motion, the nature of extra contact points with the ground removes its control autonomy and produces unexpected behavior. Based on initial field testing, construction of the Zero Speed Stability (ZSS) system was accelerated to address this concern. By implementing a set of support wheels that leave the ground once the Autocycle has reached a specific velocity, the tests of the control system could proceed.

The basic design concept of the ZSS seen in Figures 4.10 and 4.11 did not change significantly over the course of its validation and design iteration. In essence, its main components remained the same: a metal frame to mount the system to the back frame of the bicycle, a set

of linear actuators, and a set of 3D printed adapters that connect the actuators to the frame in a triangular configuration in addition to wheel mounts.

Figure 4.10: One half of the ZSS system constructed and ready to mount onto the Autocycle.

Figure 4.11: The first prototype design of the ZSS mounted onto the Autocycle.

What did change over the course of future testing was updating various components to improve reliability and robustness. The original wheels had to be replaced with larger, sturdier tires to resist frequent impact and vibration loading from field testing. The linear actuators also had to be replaced with more reliable models after the original actuators were destroyed during testing. Ultimately, the ZSS became a vital system to conducting frequent field testing without the need for an imparted initial velocity on the bicycle or some alternate control scheme.

4.2.7 Belt Tensioner

The final system that required iteration after the results of initial field testing was the torque motor belt tensioner. This system had already been iterated on, but this time, instead of a more elegant design, it simply needed to be constructed from a stronger material than PETG plastic. Using an on-campus machine shop, the team milled a new belt tensioner from 1018 steel (Figure 4.12), which would ensure that the tensioner would no longer suffer from bolt tearout or other modes of failure that had been experienced during frequent field testing.

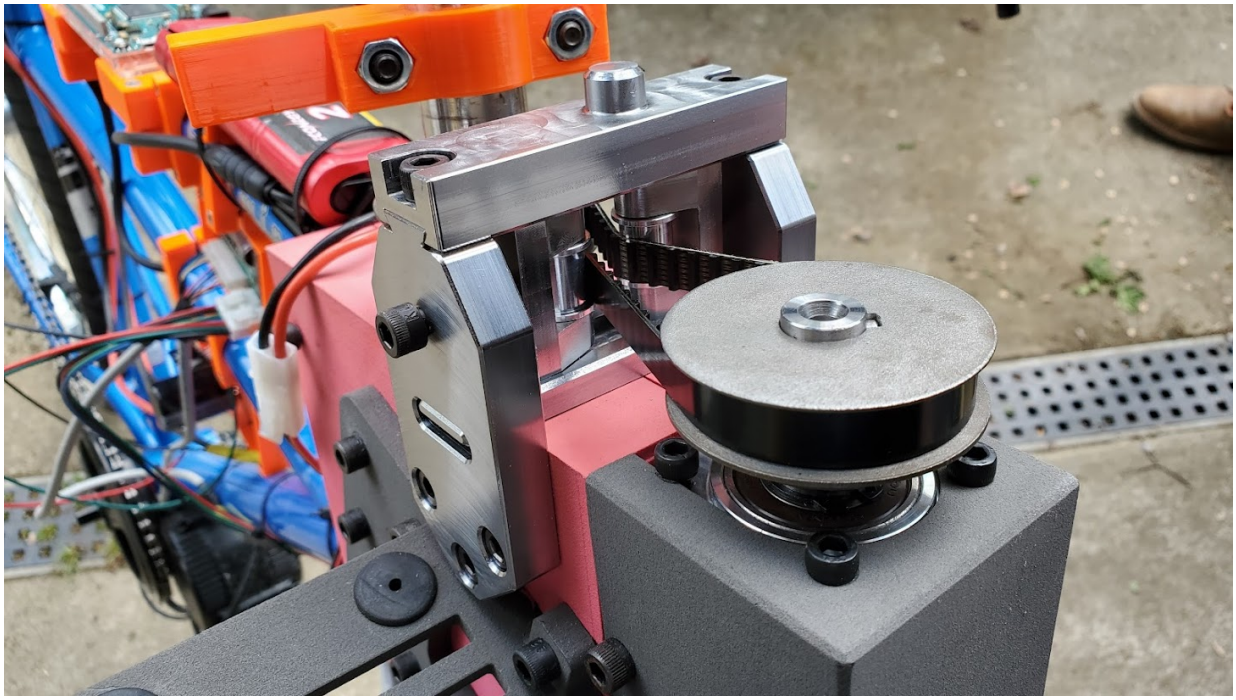


Figure 4.12: The new belt tensioner, milled from steel and mounted on the Autocycle.

4.2.8 Field Testing and Hardware Upgrades

At this point, several key milestones had been achieved by the team in the construction of the Autocycle that would ultimately lead to successful upright motion. With systems like the

ZSS, on-board data logging, and radio communication in place, the Autocycle could now operate remotely and attempt control within an environment similar to the one modeled in the control loop. Construction of a steel belt tensioner, as well as simplified electronics mounting and the implementation of a custom control printed circuit board and power distribution board, edged the Autocycle closer to a robust, repeatable testing scheme that would not require weekly repairs. By the second half of 2021, the team was ready to recommence field testing with the goal of achieving stable upright motion of the Autocycle.

Determining an appropriate location to test the Autocycle was an important priority. The team sought a location that was level, without major obstacles, and large enough to support at least 100 feet of motion in a single direction by the Autocycle. After a set of tests took place on a parking lot which had a slight slope to it, the team determined this would not be sufficient for testing, at least until a proper control scheme had been determined. The team settled on Lot 11 on the University of Maryland campus, a location which typically remained free of obstacles during weekend testing, and was sufficiently flat.

Initial testing faced difficulty from faulty actuators driving the ZSS system. The actuators were manually repaired several times during this phase of testing, later to be replaced entirely by a more robust option. Additionally, it was soon determined that the height that the ZSS tires reached when retracted may not be sufficient to allow the bicycle to achieve stability. Since it needs to turn into its fall to catch itself, the team was concerned that it needed more clearance to achieve this action. As a result, the ZSS geometry was redesigned to achieve higher lift, seen in Figure 4.13.

Figure 4.13: The revised ZSS geometry resulted in higher clearance provided by the wheels, allowing the Autocycle to tilt at least 30 degrees without contacting the ground.

The team also designed and implemented a brake system consisting of a stepper motor which pulls the handle brake (See Sec. 3.5). A larger lever arm was attached to the handlebar to minimize load necessary to brake, and to make it easier to manually pull the brake if needed during testing. The original system shown in Figure 4.14 was later redesigned to shield the wire spool from impact upon falling.

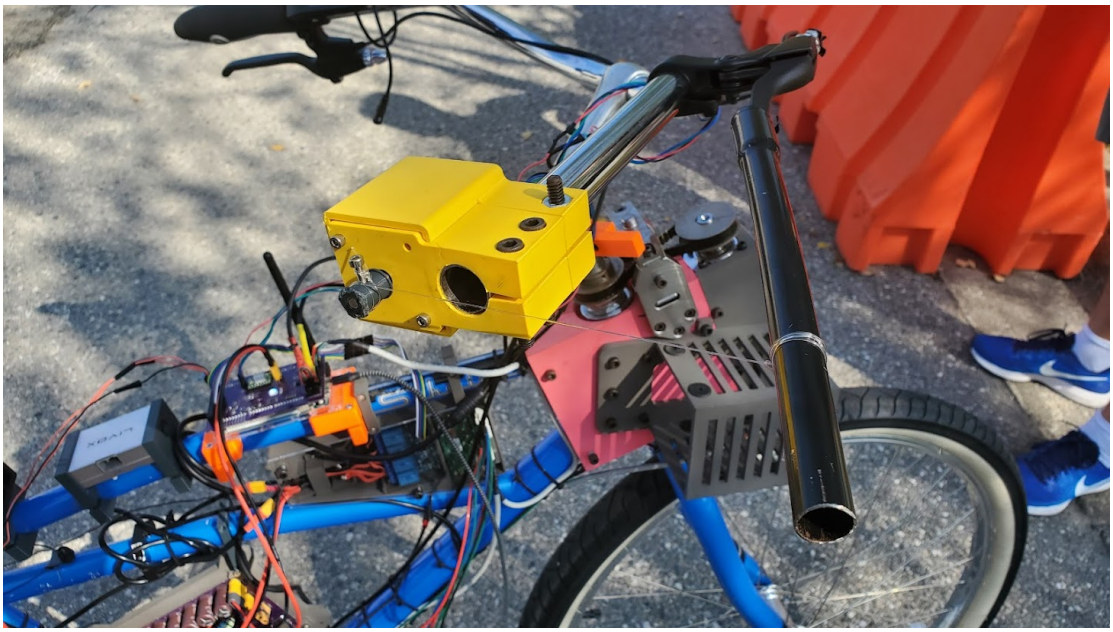


Figure 4.14: The original prototyped braking system located on the Autocycle's handlebars.

4.2.9 System Parameter Identification

As testing progressed without achieving upright motion, it was determined that even if the control scheme implemented was theoretically sufficient, the Autocycle would still fail to operate as expected unless each important input and output parameter behaved as expected. This sparked a campaign of verifying the accuracy of sensors for all critical quantities, most importantly bicycle velocity, output torque from the steering motor, and bicycle frame moments of inertia.

Previously, bicycle velocity had been estimated using the drive motor's internal encoder, however velocity data could only be reported at about 1 Hz, and the resolution of the velocity data was lacking. Additionally, the team verified through simulation that the true velocity must be within less than .5 m/s of the measured velocity to be able to achieve stable control. To rectify this, the team purchased and implemented an optical rotary encoder which was mounted to the Autocycle's chain (Figure 4.15). After some validation testing, the team was able to show that the

encoder successfully reported velocity data at a significantly higher refresh rate and resolution.

Figure 4.15: Using a 3D printed mount, the rotary encoder is connected in line with the Autocycle's chain to report accurate velocity readings.

The output torque from the steering torque motor was also measured, using a torque wrench at several set values. The team determined that the actual output torque was less than the reported output torque by a constant factor, and this offset was taken into account for the purpose of controlling the bicycle.

Finally, system identification was performed to estimate moment of inertia parameters of the Autocycle using recorded response data from previous stability tests. Using this method, each test contributed to the overall moment of inertia estimate, which proved ultimately more successful than the previous method of measuring each component using a torsional rod.

4.2.10 Further Navigation Field Testing

Initial object detection field testing consisted largely of validating object detection and path planning algorithms in non-moving, static environment conditions. After some iteration, these

efforts proved largely successful.

Figure 4.16: An example of the object detection validation process where the detected objects are plotted and compared to the environment.

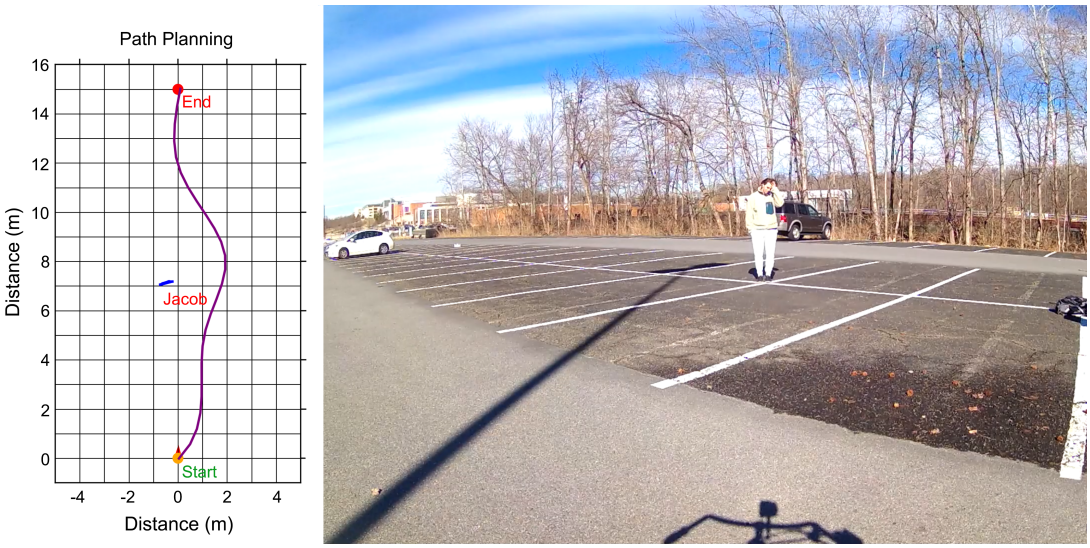


Figure 4.17: An example of the path validation process where detected objects and paths are plotted and compared to the environment.

This testing phase took about 2 months to get the object detection system and the path planning system to work first independently, and then together. The next step in testing was to validate that the bicycle could follow the path generated. To test only this capability the Autocycle uses only a ‘snapshot‘ to generate a path. Typically, the Autocycle generates new paths as it gets new information and moves along the route. In snapshot mode, a single point cloud is generated

before the test and a path is planned around any obstacles detected. In this mode of testing, the Autocycle does not generate any subsequent paths and simply tries to follow the generated path. This path was then converted into a series of steering angles as described in 3.11.6. Once this was done, the bicycle was told to move and was fed the calculated steering angles. If the environment changed in any way during the test, the bicycle would not have reacted. This test validated that the bicycle could follow a path given that the path does not change over time.

Further testing for the navigation sub-system required testing whether the bicycle could create new paths dynamically to account for new obstacles or changes in the environment At this point in time, the path planning algorithm was using the bicycle centered coordinate system described in 3.11.6. This system was able to successfully produce paths dynamically, but we were unable to resolve issues with obstacle position updating. The successful path generation was not particularly useful if the position of the obstacles were not correct after the bicycle started moving. This led to the first major revision of the path planning algorithm where the coordinates were changed to be global where the origin is the initial position of the bicycle. This looks largely the same as 4.17 with the exception that the start position can be not $(0, 0)$.

One major requirement of this new coordinate system is knowing where the bicycle is in space at any point in time. This capability was provided by the on board GPS system. The algorithm used to convert successive latitude/longitude measurements to changes in bicycle position in meters is described in 3.11.5. The GPS and the coordinate conversion algorithm both had to be validated before the rest of the path planning algorithm could be tested. To facilitate this and further validation, a data analysis suite was developed using the python library *pygame*. This suite allowed us to scrub through the timeline of any given test and see what paths the bicycle calculated as it was moved and detected new obstacles. This program also allows us to interact

with the Google Maps API to check the GPS values over the course of the test. Finally, the program also allowed us to validate the coordinate conversion by comparing the bicycle positions calculated and comparing that to the plotted GPS values.

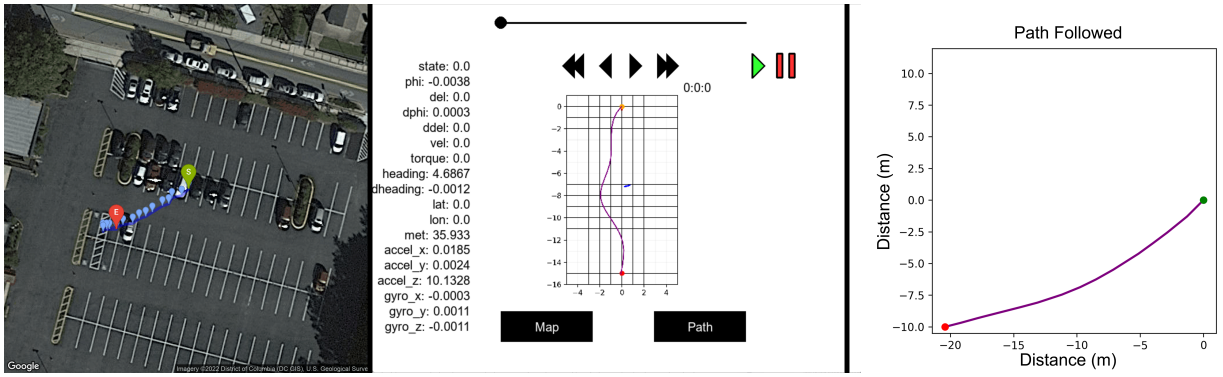


Figure 4.18: The center image shows the main screen which allows the user to scrub through the test timeline. Pressing the 'Map' button displays the image to the left which shows the GPS positions polled throughout the test. Pressing the 'Path' button displays the image to the right which shows the path followed via the coordinate conversion from the latitude/longitude points.

After several weeks of debugging the new coordinate system, we were able to validate that our GPS coordinates and our coordinate conversion looked good. Furthermore, we were able to provisionally validate that given a predefined object, the bicycle could dynamically create paths and follow them to some end position. Crucially, it was not detecting new objects during this test. The bicycle was able to keep following paths even if they had to be reevaluated due to the bicycle straying from the initial path. This could not be done with the bicycle centered coordinate system. We hedge this statement because a successful test was followed by a failed test with similar conditions. We believe the failure was due to error in the latitude/longitude measurements since the test was occurring indoors, though further testing is needed to validate this hypothesis.

At this point, the navigation sub-team was looking to start shifting from providing steering angles to the bicycle to sending desired headings. This change resulted in a number of new

challenges and realizations about the state of the navigation system. The first major problem was that in order to get a good heading values from the GPS, the bicycle needed to be going faster than than a small threshold velocity value. Getting to this velocity took time and by the time the velocity was above threshold, in order to get to the predetermined end position, the Autocycle would have to turn around. This could be dealt with placing the end point farther away, though we are usually constrained by available space in parking lots. The second issue we ran into, which stems from the first, is that when the bicycle finds itself in a position where it needs to make any kind of extreme or strange maneuver, the paths the A* graph traversal algorithm can generate paths that are impossible to the bicycle to follow given its physical constraints.

Figure 4.19 shows such a situation. In this example, the generated turning radius is about 1.5 meters. Using equation 3.4, we find that at a velocity of 2.5 meters per second, we would need a maximum steering angle of 0.823 radians, or worse at a speed of 4.5 we would need a steering angle of 0.936 radians. Both of these steering values would be rather extreme. During the test, the steering angles were limited to 0.26 radians resulting in the bicycle being unable to follow the path and the path planning algorithm would constantly reevaluate the path each of which were equally invalid, causing the program to effectively crash.

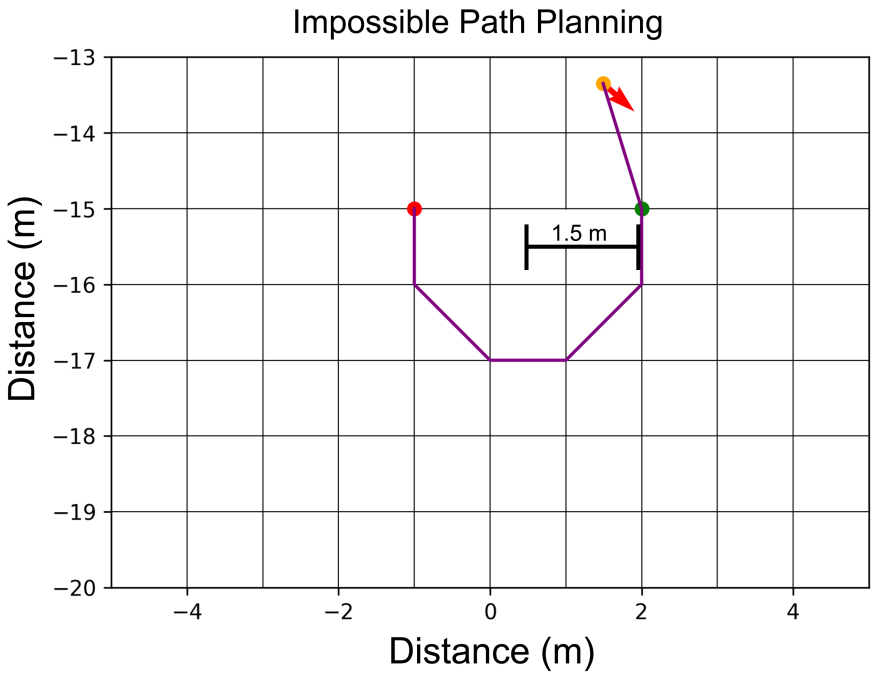


Figure 4.19: The bicycle finds itself next to the end point, but facing away from it. It must turn around if it wants to reach it. However, the turning radius of the generated path is too small.

This realization resulted in the final revamping of the path planning algorithm. In this case the change was not to the coordinate system, but to the way that the graph traversal algorithm chooses subsequent nodes. The details of this change are explained in 3.11.6. In short, this new version constrains the turning radius of the path depending on the maximum steering angle as well as the velocity of the bicycle at that point. This system, however, currently awaits validation.

4.2.11 Success: The First Upright Stable Motion

After about one full year of testing and design iteration, the Autocycle had a robust design that survived falls and a control system which had been verified in simulation, as well as system parameters which were all being measured to sufficient accuracy for testing. This began showing quickly in tests, as it seemed that all which remained was determining the proper set points for

eigenvalues and gain in the control system to achieve control.

On December 5th, 2021, the third test of the Autocycle that day was the first to achieve stable upright motion. It traveled for approximately 100 feet across the Lot 11 parking lot, and would have been able to travel even farther if there had been more space to travel. This test was repeatedly verified later that day and in future tests, demonstrating a robust control scheme able to control a bicycle's upright motion. The only major stability issue remaining to test was a graceful exit from stable motion, which was later resolved by allowing the bicycle to redeploy the ZSS and brake without falling over. An example of successful test data can be see in 4.20 and 4.21.

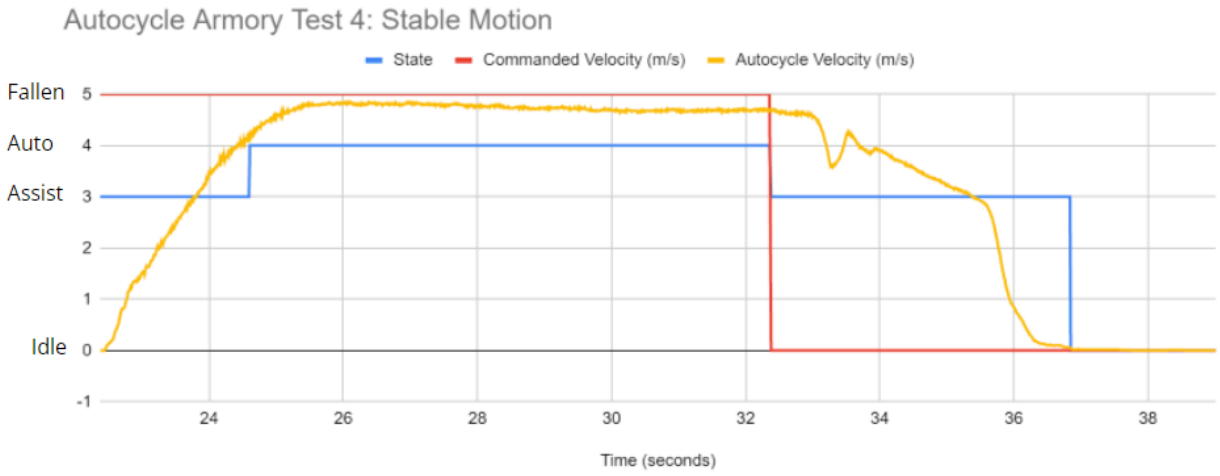


Figure 4.20: Recorded data from a test of the Autocycle that resulted in upright stable motion. The Autocycle's state remains in the automatic regime from approximately 24.5 to 32 seconds after test initiation.

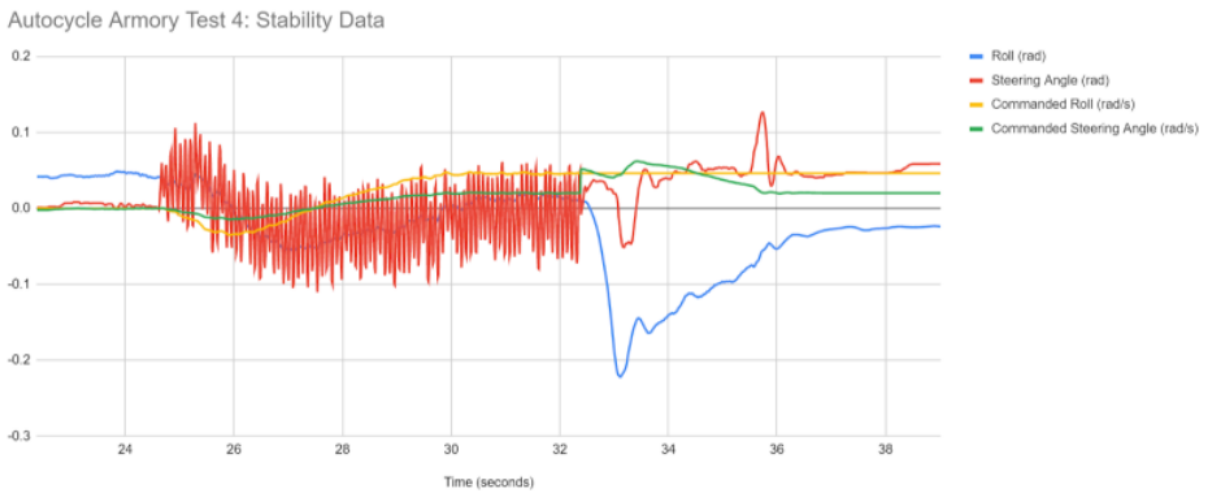


Figure 4.21: More data from the same test as 4.20, demonstrating the behavior of the roll and yaw of the Autocycle during stable motion.

4.2.12 Heading Control

The next stability goal to achieve was to not only control the lean of the Autocycle, but also its overall heading. The natural tendency of the Autocycle without heading control is to swerve left or right to stabilize itself, often veering far off of a straight course. However, with heading control, the bicycle could travel in a straight line, or, eventually, make designated turns.

Using the interior of the Armory gym on the University of Maryland campus, the team successfully demonstrated heading control across 150 feet of stable upright motion. The heading control showed a marked improvement in matching a specified heading over the same control scheme merely attempting to keep the bicycle upright, seen in 4.22.

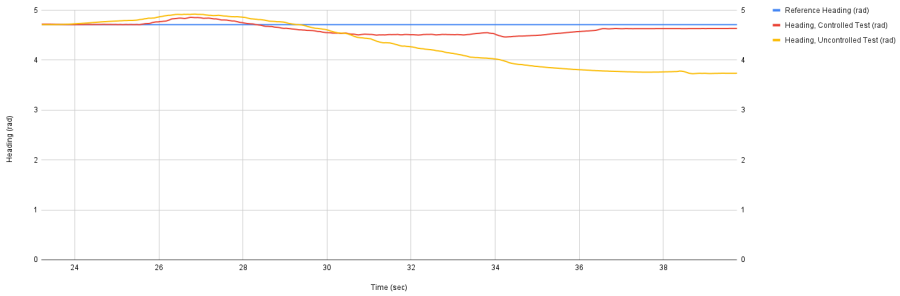


Figure 4.22: The heading of the Autocycle compared between an uncontrolled and controlled test.

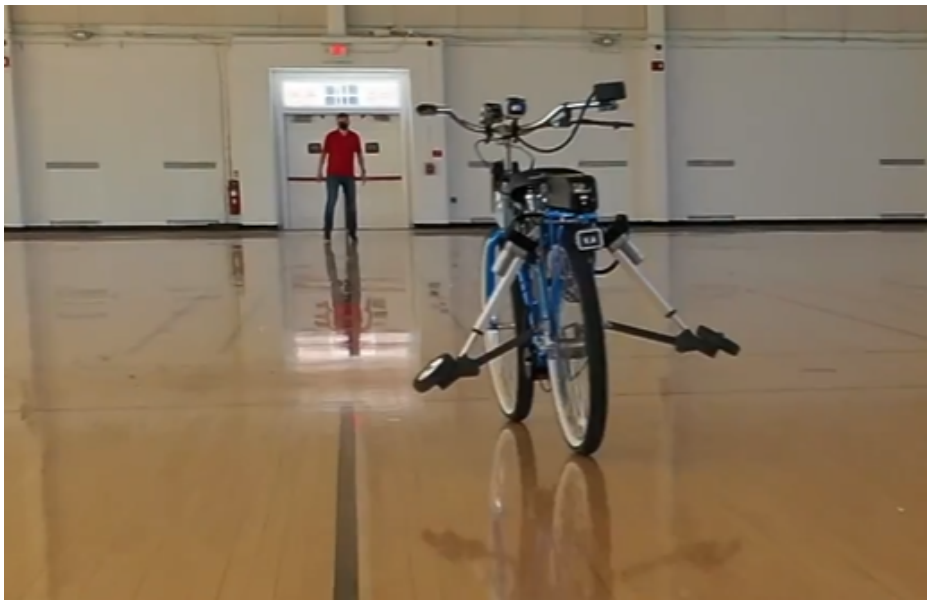


Figure 4.23: The Autocycle travelling under its own control across the Armory gym.

Chapter 5: Pitfalls and Challenges

5.1 Bike Frame

A recurring set of problems run into by the team was due to our selected bike frame. The Schwinn Cosgrove bike frame was initially selected due to its low cost and wide wheels, however the frame is not standardized. The middle tube through the center of the main triangle was useful at points as an additional mounting location, but it often interfered with component placement. Another challenge posed by our frame was the attachment of the drive motor. The selected Bafang motor was designed for a standard bicycle frame bottom bracket, but the Cosgrove bicycle's bottom bracket was too wide. To rectify this, an angle grinder was used to cut down the width of the bottom bracket to allow the motor to fit. The rear brake mount also became an issue when creating the braking assembly. The mount on the frame provided a single hole for mounting brake calipers, the standard for road bikes, however the road bike calipers we purchased are not large enough to fit around our wide wheels. This necessitated manufacture of a custom bracket for the brake calipers that converted the one hole on the frame into the two screws on our brake calipers.

5.2 Controls Inexperience

When the team first formed, no members had any formal knowledge of control systems beyond some examples from linear differential equations. This meant that the team initially lacked the context and background to navigate much of the literature. However, thanks to the help of able mentors and a good deal of self-teaching, the team developed its first state space controllers and estimators before any members had taken a formal modern controls course, and only advanced from there.

5.3 Delays

5.3.1 Sunny Days

Throughout the testing of the bicycle, the on-board LiDAR system frequently encountered interference issues on days when the sunlight was strong enough to interfere with the sensor. This caused severe issues where obstacles were either not detected, or obstacles were spotted where there was nothing but open air. This caused a number of delays, especially during summer months. Livox has since released an automotive grade LiDAR that blows the Horizon out of the water at most specs, unfortunately at the time of purchase there were not many LiDAR options within the budget that could fit the team's needs. As the technology develops, the entry barrier for LiDAR will hopefully continue to drop.

5.3.2 Testing Without the ZSS

It was a common occurrence during early stability tests that the control system would cause the bike to roll to one side and collapse. Sustained falls, vibratory loads, and fatigue on components due to strenuous testing would cause ZSS components to break off, requiring replacement and repair that could take weeks. Without the ZSS, it would be impossible to conduct navigation tests, as the Autocycle would not be stable enough to remain upright while following the path laid out by path planning algorithms. Similarly, while the ZSS was in disrepair, stability testing would be impossible, since the bike needed to start from rest and accelerate through a region of unstable speed until it reached the minimum threshold velocity for the active control system, at which point the damaged ZSS would not be necessary. Damaged actuators were especially time consuming, as new ones would need to be ordered and shipped within the week in order to make testing deadlines. More reliable actuators, higher quality 3D printed components, sturdier rotation brackets, and Locktite on screws to minimize vibratory loads ensured that the ZSS would sustain less damage, allowing for more testing to be done in between maintenance. More about the ZSS actuators can be found in 5.4.

5.4 Broken Parts

5.4.1 Tensioner Assembly

The torque motor that controls the actuation of the handlebars required a belt and pulley interface in order to transmit torque to and change the orientation of the front wheel. The adjustable tensioner assembly placed on the torque motor mount, when tightening the belt along

the driving and driven pulleys, was highly susceptible to damage during testing. Early iterations of the tensioner design were fabricated out of PLA plastic, using a 3D Printer. Whenever the prototype failed to stabilize during an outdoor test, the tensioner ran the risk of violently making contact with the ground. This resulted in both disassembly and damage to the tensioner. Multiple tensioner assemblies have been broken over the course of testing, resulting in the decision to fabricate the final tensioner out of steel. The steel tensioner was more impact-resistant, and better fit with the overall aesthetic presentation of the Autocycle. The steel tensioner was treated with a cold bluing solution to keep it from rusting, and the final tensioner assembly has been reliably used since May of 2021.

5.4.2 ZSS Actuators

Another major source of breakages were the actuators that controlled the raising and lowering of the ZSS wheels. They were chosen primarily to give as large of an extension length and speed as possible, as it was considered important for the ZSS to have the greatest possible swing. There was also a concern that the control system might encounter problems trying to function while the ZSS was partially deployed, unfortunately the overlap of these two desires proved to be small, leaving few options and alternatives to choose from when it came to picking actuators. The decision was eventually made to acquire a set from Progressive Automations. The first set of actuators ordered turned out to have been improperly specced, and fused their brushes immediately upon activation. Once the proper electrical adjustments were made and new actuators were installed, durability issues began to crop up. The next set of actuators began to fail when a brush cracked under a minor impact, necessitating a replacement. Then electrical problems

began to crop up again, in spite of previous fixes. Investigation of the actuators' internals was unable to determine a cause of these issues, despite several attempts and the replacement of some electronic components. In the end, it was determined that the actuators were from a faulty batch, and that any replacements would also come from the same batch. The problem was eventually resolved by redesigning the entire ZSS to support a different model of actuator that sacrificed speed for positional control and greater reliability, but not before six actuators were destroyed and/or scrapped and used for parts.

Figure 5.1: The brushed motor retained its graphite brushes inside of a plastic housing. It was a common failure for this housing to melt, and render the brush unusable.

5.4.3 ZSS Frame Components

The ZSS frame itself required numerous modifications over the course of testing, due to part failures and necessary modifications needed to incorporate new design elements. The brackets that originally held the linkage arm and linear actuator were made of aluminum with threaded

hole that aligned with a hole pattern on the steel frame. Manufacturing defects with the threaded holes resulted in misalignment that caused bending and vibrations in the actuator and linkage arm, impacting the stability of the prototype during testing. Moreover, vibration and impacts with the ground would often cause the screws that retained the brackets to the frame to loosen and fall out, requiring immediate repair. These brackets would be remade multiple times to address the fit of bolts. A seated bearing inside of the bracket would be used to ensure that the shoulder bolts retained the linkage arm and actuator. Alignment of the brackets into the same plane proved challenging due to the low depth of the threaded holes that retained the brackets, so spacers were used generously to align the ZSS components. Ultimately, this system required constant maintenance, and was challenging to remove for maintenance, so the brackets were replaced with purchased brackets and shoulder bolts that were compatible with the Frigelli Automations' linear actuators with minimal modifications.

Adding to the troubles with the early models of the ZSS, the design of the frame in which the actuators sat was also problematic. It turned out that if overextended, the actuator could form a knee joint with the rest of the frame, preventing it from being retracted. It is possible this was responsible for the destruction of at least one actuator, which might have failed attempting to force this point, although this could not be confirmed and may simply have been the electrical faults determined to be present in the actuators. This issue was addressed by the addition of two mounts that held limit switches such that they would be closed and cut off power to the actuators when they reached their intended extension. However, this created a new point of failure- the limit switches and their mounts were fragile, and often broke during testing, necessitating test cancellation for the day so they could be repaired. This issue was resolved with the second set of actuators as they possessed servo positional control, removing the need for the limit switches, and

the frame was additionally redesigned to reduce the likelihood of such a joint forming to begin with. This was not the end of the troubles with this system, however. The new design featured bearings to hold the lower arms in place, which quickly proved prone to wobble and meant that the part could shift significantly backwards and forwards in response to ground obstacles. This was eventually fixed by welding the bearings to their associated bars, which solved the issue.

Figure 5.2: The aluminum brackets used to retain the linkage arm and linear actuator required customized spacers to keep the components of the ZSS in plane.

The housing for the training wheels, which served as the connection point for the linkage arm and linear actuator, was a component that needed to be changed multiple times due to damage sustained during operation, and changes to the design methodology. It was discovered that PLA plastic was insufficient for surviving the loads placed on the wheel mount during operation. The first prototypes broke apart, leading the team to design new components out of nylon, 3D printed in the instructional fabrication laboratory in Clark Hall. These nylon components were resized

and remodelled when it was discovered that their weight was creating a greater moment on the mounting brackets and their mounting screws when they would vibrate during testing. In the interest of minimizing material cost and weight, these were redesigned to be smaller and shorter.

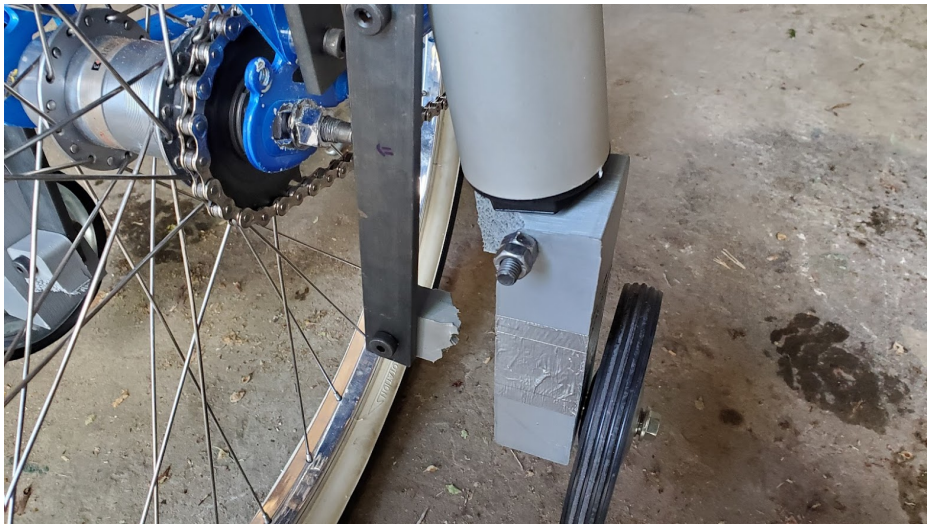


Figure 5.3: PLA proved to be too weak to sustain the loads placed on the block during actuation. Future designs would be fabricated out of nylon.

5.4.4 Rear Wheel Coupler

One of the earliest troublesome parts on the bike was the rear wheel coupler, which was an integral part of the internal gear shifting mechanism the bike originally came with. Although the shifting capability was disabled, the part itself remained vital to the structure and function of the wheel hub. This quickly became a problem, as the original part could not handle the stress the motor applied to it, and it quickly broke during testing. A replacement was produced using 3D printing, which also broke shortly into testing. The next replacement was printed using carbon-fiber reinforced Onyx and managed to hold up for several months before failing. The final version was again printed, but even more heavily reinforced, and was able to endure testing.

5.4.5 Drive Motor

The first part to break was the drive motor itself. It used a non-standard method of communication: it used its own custom UART protocol meant to talk only to its monitor and throttle. Additionally the documentation for the motor turned out to be severely lacking, meaning that information regarding the pinout and communications protocols was incomplete and had to be worked around. As this process was ongoing, the motor we were using failed and could not be repaired. The exact cause of this failure is still unknown; it is suspected that the full 48 volts from the battery were applied to the data input of the motor, destroying the motor controller in the process. This was a severe setback, however a temporary motor was procured, and testing was able to continue with it until a permanent replacement could be found.

5.5 Performance Issues

5.5.1 LiDAR

The point clouds obtained from the LiDAR included points that could not exist: y values (vertical axis values) that would result in points being underground. Our hot-fix to the issue was to filter out points found to be considered as underground which was set based on the LiDAR's height relative to the ground. In practice, this solution creates some problems and edge cases such as if the bicycle is looking out over the crest of a hill. This situation would result in most if not all points being filtered out resulting in data lost until the bicycle is level with the decline. In future research this hot-fix could be avoided through filtering based on having more accurate point cloud or through filtering based on point density, something looked into but not implemented as

a greater point density overall was needed for this to be an accurate fix.

As stated in the delays section above, the LiDAR used was barely within its rating for maximum environmental brightness, making the data have major inaccuracies. These performance issues could be solved in future research endeavors through use of a higher performance LiDAR, but at the time of LiDAR selection the available options to our team were not sufficient for the Autocycle.

5.6 Impact of COVID-19

5.6.1 Shipping Delays

Although the advent of COVID-19 had a substantial impact on the construction of the bike, workarounds were quickly found that allowed the work to continue in spite of the delays and setbacks it caused. The hardest of these issues to mitigate were the delays caused by the global logistics breakdown. Ordered parts sometimes arrived weeks behind schedule, stalling an entire avenue of progress. The ZSS was an excellent example of this; replacement actuators were especially delayed, which simultaneously delayed the progress of the entire stability subteam and navigation subteam.

5.6.2 University Closures and Testing Locations

In addition to the slowed shipping, the closure of the University was a serious challenge to overcome. However, this one proved to be far less severe than the former issue, thanks to the fact that many of the University's machine shops and fabrication labs remained open, even in the teeth of the pandemic, ensuring that the manufacturing of the many custom parts used on the bike

could continue in spite of the closure. Terrapin Works and the University's Instructional Fabrication Lab both proved instrumental in this, allowing for experienced personnel to manufacture complex parts and inexperienced ones to learn the basics by manufacturing simple ones. This was supplemented by the team's private 3D printers, which could be used to produce many of the parts with little difficulty and minimal contact with the outside world.

The closure also forced the relocation of the prototype, which was housed in a University facility at the time. However, an alternate construction and testing location was quickly found, and in fact proved advantageous due to more favorable location to some personnel. That being said, the pandemic did hinder efforts to find good testing locations. Obtaining the use of a properly sized space for prolonged tests of a vehicle that must travel at five meters per second to maintain its balance proved very difficult. Tests were conducted in various sizes of parking lot and roads to varying degrees of success, while the pandemic made it difficult to work with the University to acquire a suitable permanent testing location.

Chapter 6: Discussion & Conclusions

6.1 Comparisons to Other Scholars

There are two main comparable projects that were kept in mind during work on the prototype. The first was a team of undergraduate students from the Indian Institute of Technology Karagpur (IIT) [12]. Their model used a pair of spring-driven training wheels to maintain the bicycle's balance and a pair of ultrasonic sensors for obstacle avoidance. It was able to avoid obstacles and self-balance, but was limited to low speeds and relatively smooth terrain due to their passive stability system. In contrast, the prototype Autocyle was able to drive much faster, running around 5 m/s. Autocycle features a more sophisticated LiDAR-based obstacle avoidance system that promises greater range and detail, but in practice was difficult to implement. The other comparable project was a team of three Chinese researchers from Tsinghua University [13]. Their paper focused primarily on obstacle avoidance, but their bicycle was also capable of self-stability. It was fairly slow, operating primarily at 2.2 m/s and only over smooth ground, but it possessed a very advanced computer vision system, even able to identify and evade puddles of water on the ground. Its vision capability far outstripped those of the Autocycle prototype, but the Autocycle is far faster and more robust on unstable or uneven ground. Overall, the Autocycle proved more capable of high-speed balance than its compatriots, but generally less capable at identifying and locating obstacles. The Autocycle prototype's vision system is more prone to

errors than the IIT team's bicycle and simply less effective than the Tsinghua University team's bicycle.

6.2 Future Work

6.2.1 Continuation

While the project has met our personal goals, there are still some steps that can be taken to continue to improve it. Integration between the navigation and stability sections is still ongoing, due to the risks inherent in conducting stability tests (which tended to fail violently) with the sensitive, vulnerable, and expensive navigation components on board. In spite of these challenges, the current prototype has begun to suffer from failures due to oversights during design decisions made early in the design process. Early design decisions tended to maximize single design parameters. Some of the decisions surrounding our design parameters seemed sound initially, but have caused unforeseen issues during the development of the rest of the prototype. Similarly, underestimating the physical and computational requirements of the bike early in development often led to the selection of functional, yet sub-optimal components. An example of the former issue: the bike frame was selected for the widest standard tires possible, to aid with balance, however the selected bike frame proved to have a very non-standard design, requiring extensive machining and modifications to the frame in order to mount a drive motor and other components.

Though the width of the frame was desirable, the twisting, curving surfaces that make up the frame were unsuitable for mounting large components. As a result, the battery was mounted exposed on the back carriage, rather than mounted between the members of the frame as was initially designed. An example of the latter issue was the selection of an Arduino Due to serve

as the controller for the bike's stability system. When the Due was considered for purchase, it was deemed overpowered for the needs of the bike; Smaller and less powerful controllers were similarly considered, including the Arduino Uno, a similar, but less powerful board, however as the control system grew in complexity, the Due became more and more necessary. With the current control system, the Due reaches near the limits of its abilities, largely due to the lack of an onboard floating point arithmetic unit. Compounding this issue is the obstacle avoidance system. When fully implemented, it would use a camera and LiDAR system, which needed to feed data into a separate controller, a RasPi, which processes the data, calculates a path, and uses that path to issue steering commands to the bike. Further research has discovered that a single smartphone could replace all of these systems more inexpensively while offering more processing power. An improved version of the Autocycle would surely require a ground-up redesign, featuring a different bicycle as the chassis, a more powerful controller, a more standard drive motor, a more versatile and powerful vision system, and many other improvements.

6.2.2 Bike-sharing System Integration

To complete the initial bike-sharing system model, several additional features would need to be implemented. On the user end, an app would need to be designed that would allow a user to call a bike. An app like this would likely feature a user interface that works similarly to apps like Uber. On the logistics end, charging stations compatible with the Autocycles would need to be designed (which would likely require a total redesign of the prototype), as well as specialized repair centers for the electronic components. The bikes themselves would be far more costly than a normal bicycle, due to the comparatively expensive electronics and actuators that would need to

be added, however, they would offer some substantial advantages over a traditional bike-sharing systems. The ability of the bike to make its way to customers independently is an obvious defining feature, but arguably more useful would be the ability of the bikes to redistribute themselves to hubs when necessary. Much time and effort is spent by bike-sharing system companies moving bikes by truck to ensure they are where they need to be for the next day's riders. Even worse, problems can occur when they are unable to or choose not to bother redistributing bikes; they can quickly pile up in the hundreds, and may eventually be scrapped; transported to fields or waste dumps full of their abandoned brethren to rust. The Autocycle would be able to avoid these problems as they could redistribute themselves during periods of time when they are not be needed.

6.2.2.1 Human Usability

An actual human-usable version of the prototype would also require some redesign. The current version cannot mount pedals due to the encoder mount interfering with their movement arc, has exposed charged capacitors that protrude into space where a user's legs would need to pass through, and faces difficulties attempting to implement to manual braking due to the mount for the brake motor sitting where a human user's hand would go. The redesigned version would need a more elegant and streamlined system for mounting components within the frame, an autonomous braking mode integrated into the brakes themselves rather than being forced to actuate the brake handle, and a re-positioned encoder mount.

6.2.3 Alternative Uses

Finally, there are some alternative uses for which the Autocycle could be suited. Much infrastructure is already prepared for the use of human-operated bicycles, meaning Autocycles are better suited for city use than most other robot configurations. They could easily see use for light item transportation, high-priority online delivery, or other similar processes. It is also theoretically possible for the bike to feature a riding assistance mode, where the bike's control system could aid a human rider if necessary, although this would require extensive calibration and safety testing that might be hazardous to human test subjects.

6.3 Concluding Thoughts

All said, while the Autocycle does not represent a ground-breaking leap forward in controls system design or robotics, it serves to demonstrate what is necessary for a practical, self-balancing, autonomous bicycle. It has been able to go from a full stop up to speed, self-balance, and brake back to a full stop consistently in testing, even under sub-optimal road and weather conditions. It has also demonstrated the ability to avoid obstacles and path plan effectively. It is capable of moving quickly, it is compact, and it can safely navigate and balance itself. We hope that our work is built upon in the years to come to make autonomous bicycle roaming the streets a reality we can all enjoy.

Appendix A: Equity-Impact Report

Throughout the life of the research project, the general goal is to focus on collecting results that will further human knowledge. However, in all well-developed studies, every step of the process starting from the idea creation phase needs to be carefully constructed so as not to introduce any inaccuracies, bias, or other shortcomings that could undermine the success and credibility of the project and the research team. More recently, any well-developed research also needs to be designed in a way that ensures equitable practices are being used throughout its lifespan. In the preceding report, team Autocycle showed that the research that has been conducted over the past three years has been done so appropriately and equitably.

The purpose of this research is nothing more than simply creating a self-balancing, self-driving bicycle that can be used in bike-sharing systems. Aside from the fact that the project's scope is limited to areas that can sustain bike-sharing systems (e.g. cities), the results in the above paper do not explicitly benefit any particular racial or ethnic group(s). In an article by Brookings, analysis of the 2020 US census demonstrates that the 50 largest US cities are becoming increasingly more diverse over time [61]. Thus as cities grow more diverse, team Autocycle's conclusions will apply to an increasingly more diverse group of people over time. Looking further than just US cities, this research has the potential to impact any city or area across the globe that's able to sustain bike-sharing systems. Aside from looking at different racial and ethnic

groups, Autocycle's research also has the potential to positively affect people from across the wealth spectrum. Since bicycles are much less expensive to maintain than traditional vehicles, a bike-sharing system made up of self-driving bicycles is extremely accessible to whoever wants to use it, no matter their income status. This idea would be able to efficiently mobilize all citizens living in an area with bike-sharing systems. The only group of people not discussed so far are those living with disabilities that may not be able to ride a physical bicycle. Because of the Autocycle's riderless abilities, variants of the base design could support apparatuses that can pull those who are physically unable to ride a bike, almost like a horse-drawn carriage.

Since team Autocycle's research is only focused on the proof-of-concept of a self-driving, self-balancing prototype, passion for the project is the only requirement. Each team member brings an important perspective to the concept of the bicycle, and the resulting conclusions throughout the semester are the culmination of the group's ideas. Although Autocycle's members are all male and mostly white, the robust engineering design process that has been employed throughout the entire process should be sufficient to eliminate any unintended bias. The team also consulted with a diverse group of experts during major milestones to obtain increased breadth and depth of ideas, knowledge, and inspiration. Any research into urban implementation, however, would require further consultation with professionals in that field.

The research that team Autocycle is working on benefits people who live in urban areas or areas that can maintain bike-sharing systems indiscriminately. As previously mentioned, the cost of bikes is significantly less than other methods of transportation, so everyone who wants to be able to use a system made up of self-driving, self-balancing bicycles can do so easily. Additionally, the Autocycle can be modified and outfitted to transport people or goods that would not normally be able to ride a traditional bike. The intended consequence of this research is to

be able to make bike-sharing systems more accessible. This way, more people are encouraged to use bikes rather than gas-powered vehicles, which could reduce both carbon emissions and traffic in busy areas. A potential unintended and negative consequence could come in the form of too many bikes in one area; if the popularity of self-driving bikes skyrockets, it may not be as easy for pedestrians and traditional vehicles to navigate usual routes. Team Autocycle does not view this negative consequence as being likely, concluding that our research as a whole is a net positive for society. Further research on implementation in areas that can sustain bike-sharing systems should be done to better measure the impacts of positive and negative consequences.

Success for this research project is measured by understanding and completing our original goal: can a self-driving, self-balancing bike that is intended to be used in a bike-sharing system be made? This goal does not rely on the completion of a fully-working prototype, rather it seeks to measure the feasibility of such a system. Early on in the research process, simulations gave proof that, in the ideal case, initial hypotheses are viable. The metrics for success changed as the Autocycle came to reality; the testing and iterative design that has been completed up to this point shows that this system should be explored further since it is clear that a robust prototype can exist.

Appendix B: A Note on Ethics

As technological innovation skyrockets, so does the need for privacy among users and businesses. Electronic devices that track user data, location, usage, and even personal information are sometimes used by manufacturers to distribute and sell to third-party corporations. The ethical decisions made by these companies regarding privacy and data usage are often under question, and some CEOs are even brought before congress (i.e. M. Zuckerberg). This ethical understanding is bigger than just personal privacy, however. It encompasses all of the important decisions that are required to create safe and just technological innovations. As technology continues to develop, how will personal information remain safe? What factors need to be considered to keep users away from harm? What ethical decisions need to be made when creating an artificially intelligent device? What are the most important aspects of everyday life that need to remain undisclosed? And, most relevant to the issues and solutions outlined in the above prospectus, how do ethics play a role in advanced and autonomous vehicles?

Ethics, defined as the moral principles that govern one's actions, and social justice, loosely defined as the demand for responsible, culturally-informed ideals, are necessary for the creation of equitable and just technologies [62]. These two terms play a big part in daily lives, as they govern human interaction and innovation. In the realm of technology, human social relations are often shaped by new inventions and creations. Additionally, the "IEEE Code of Ethics states

that engineers should aim to improve their understanding of the technology, its appropriate application, and the likely consequences of its implementation" [63]. This statement, outlined by IEEE, guides inventors' thinking towards creation without any negative consequences or inappropriate uses. These defined terms are crucial to the understanding and evaluation of the ethics of self-driving vehicles in particular when considering the scope of this paper.

A common predicament when it comes to decisions made by self-driving vehicles is the trolley problem. Consider a trolley on a specific track. The track splits into two paths, and the trolley can take either one. However, on the left track is one group of people (whose professions and social standings are determined by a particular instance of the problem), and on the right track is another group of people (with similar parameters to the first group). The trolley cannot slow down in time to avoid hitting a group of people, so how does the operator decide to proceed? While this issue isn't reflective of an exact, modern-day situation, the ethical considerations are necessary to understand. How should a modern vehicle proceed when faced with an ethically impossible decision? There is no one right answer to this question, but it is an important step towards generalizing the impact of intelligent technology on human lives [64].

Although self-driving bikes are less dangerous than self-driving cars, many of the same principles apply. For example, in heavy traffic, how should self-driving cars proceed if someone in a car in the back of the queue is on the way to the hospital for a medical emergency? Codes of social justice guide reasoning to prioritize medical emergencies, but how should the problem differ if the operator of that particular vehicle is lying? [63]. Scaling the issue back down to bikes, how should a self-driving bike be able to communicate with pedestrians to subsequently prioritize a particular path? When building an intelligent vehicle, it is socially just to consider all factors in its immediate vicinity.

Some of the biggest ethical considerations that are relevant today are those concerning privacy. Any device with location services or access to centralized servers tracks and stores users' data, including location. This information could potentially enhance the users' experience. Previous routes, location history, etc could be used to minimize delays [63]. But the potential for ethically wrong decisions is huge. Companies often sell personal data for profit to advertising companies and other large businesses. So then, should no personal data be collected at all at the expense of the users' experience? If some data is collected, is it safeguarded from companies seeking profit?

When it comes to an ethical understanding of self-driving vehicles, in particular, privacy, communication, and appropriate creations all need to be considered. Integrity deteriorates when one is profit-motivated, so strict guidelines need to be established early on. The pursuit of a socially just, ethical innovation is necessary for a positive future, so certain protocols need to be in place to create a sustainable, long-lasting creation.

Bibliography

[1] J.p Meijaard, Jim M Papadopoulos, Andy Ruina, and A.l Schwab. Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2084):1955–1982, August 2007.

[2] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. *The Geography of Transport Systems*. Routledge, July 2013. Google-Books-ID: PfEdAAAAQBAJ.

[3] Paul DeMaio. Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation*, 12(4), December 2009.

[4] City of Wilmington. Bike Share Feasibility Analysis. `https://www.wilmingtonde.gov/home/showpublisheddocument/574/636007689764630000`. 2022-03-28.

[5] Federico Chiariotti, Chiara Pielli, Andrea Zanella, and Michele Zorzi. A Dynamic Approach to Rebalancing Bike-Sharing Systems. *Sensors*, 18(2):512, February 2018. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[6] Cyrille Médard de Chardon, Geoffrey Caruso, and Isabelle Thomas. Bike-share rebalancing strategies, patterns, and purpose. *Journal of Transport Geography*, 55:22–39, July 2016.

[7] Paul Berger. The Hottest Thing in Dockless Bike Shares: New York City. *Wall Street Journal*, June 2018.

[8] Elliot Fishman, Simon Washington, and Narelle Haworth. Bike Share: A Synthesis of the Literature. *Transport Reviews*, 33(2):148–165, March 2013. Publisher: Routledge _eprint: https://doi.org/10.1080/01441647.2013.775612.

[9] Jueyu Wang and Greg H Lindsey. Do new bike share stations increase member use: A quasi-experimental study. *Transportation Research, Part A: Policy and Practice*, 121:1–11, March 2019.

[10] U.S. Department of Energy. Electric Vehicle Benefits. `https://www.energy.gov/eere/electricvehicles/electric-vehicle-benefits`.

[11] Tesla Motors. Autopilot. `https://www.tesla.com/autopilot`.

[12] Ayush Pandey, Subhamoy Mahajan, Adarsh Kosta, Dhananjay Yadav, Vikas Pandey, Saurav Sahay, Siddharth Jha, Shubh Agarwal, Aashay Bhise, Raushan Kumar, Aniket Bhushan, Vraj Parikh, Ankit Lohani, Saurabh Dash, Himanshu Choudhary, Rahul Kumar, Anurag Sharma, Arnab Mondal, Chendika Karthik Sai, and P N Vamshi. Low cost autonomous navigation and control of a mechanically balanced bicycle with dual locomotion mode. In *2015 IEEE International Transportation Electrification Conference (ITEC)*, pages 1–10, August 2015.

[13] Mingguo Zhao, Sotirios Stasinopoulos, and Yongchao Yu. Obstacle detection and avoidance for autonomous bicycles. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1310–1315, Xi'an, China, August 2017. IEEE.

[14] Whipple. The Stability of the Motion of a Bicycle. *Quarterly Journal of Pure and Applied Mathematics*, 30:312–348, 1899.

[15] J. D. G. Kooijman, J. P. Meijaard, Jim M. Papadopoulos, Andy Ruina, and A. L. Schwab. A Bicycle Can Be Self-Stable Without Gyroscopic or Caster Effects. *Science*, 332(6027):339–342, April 2011.

[16] J. D. G. Kooijman, A. L. Schwab, and J. P. Meijaard. Experimental validation of a model of an uncontrolled bicycle. *Multibody System Dynamics*, 19(1):115–132, February 2008.

[17] Jeff Watson. MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments | Analog Devices.

[18] Kenneth Gade. The Seven Ways to Find Heading. *Journal of Navigation*, 69(5):955–970, September 2016.

[19] T. Luettel, M. Himmelsbach, and Hans-Joachim Wuensche. Autonomous Ground Vehicles—Concepts and a Path to the Future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1831–1839, May 2012.

[20] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2147–2156, Las Vegas, NV, USA, June 2016. IEEE.

[21] Yazan A. Alqudah and Belal H. Sababha. On the analysis of road surface conditions using embedded smartphone sensors. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 177–181, Irbid, Jordan, April 2017. IEEE.

[22] Yoshiaki Taniguchi, Kodai Nishii, and Kiroyuki Hisamatsu. Evaluation of a Bicycle-Mounted Ultrasonic Distance Sensor for Monitoring Obstacles and Holes on Road. *Int. J. Simul. Syst. Sci. Technol*, 16(6):1–1, 2015.

[23] Wang Zhiqiang and Liu Jun. A review of object detection based on convolutional neural network. In *2017 36th Chinese Control Conference (CCC)*, pages 11104–11109, Dalian, China, July 2017. IEEE.

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *arXiv:1708.02002 [cs]*, August 2017. arXiv: 1708.02002.

[25] Petru Soviany and Radu Tudor Ionescu. Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction. *arXiv:1803.08707 [cs]*, March 2018. arXiv: 1803.08707.

[26] Feng Yu, Zhen He, and Na Xu. Autonomous navigation for GPS using inter-satellite ranging and relative direction measurements. *Acta Astronautica*, 160:646–655, July 2019.

[27] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009.

[28] D. T. Lee. On finding the convex hull of a simple polygon. *International Journal of Computer & Information Sciences*, 12(2):87–98, April 1983.

[29] Phillip Calais. Building an electric bike. *ReNew: Technology for a Sustainable Future*, 71:37–41, 2000.

[30] Carmelina Abagnale, Massimo Cardone, Paolo Iodice, Salvatore Strano, Mario Terzo, and Giovanni Vorraro. Model-based control for an innovative power-assisted bicycle. *Energy Procedia*, 81:606–617, 12 2015.

[31] Ricardo Meireles, Jose Silva, Alexandre Teixeira, and Bernardo Ribeiro. An e.bike design for the fourth generation bike-sharing services. *World Electric Vehicle Journal*, 6:58–63, 03 2013.

[32] M. R. García, D. A. Mántaras, J. C. Álvarez, and D. Blanco F. Stabilizing an Urban Semi-Autonomous Bicycle. *IEEE Access*, 6:5236–5246, 2018.

[33] Lei Guo, Qizheng Liao, and Shimin Wei. Nonlinear stabilization of bicycle robot steering control system. In *2009 International Conference on Mechatronics and Automation*, pages 3185–3189, August 2009.

[34] Vito Cerone, Davide Andreo, Mats Larsson, and Diego Regruto. Stabilization of a riderless bicycle: A linear-parameter-varying approach. *IEEE Control Syst. Mag*, pages 23–32, 2010.

[35] T. Murakami, R. Nakamura, F. Yu, and K. Ohnishi. Force sensorless impedance control by disturbance observer. In *Conference Record of the Power Conversion Conference - Yokohama 1993*, pages 352–357, April 1993.

[36] Saeed Hashemnia, Masoud Shariat Panahi, and Mohammad J. Mahjoob. Unmanned bicycle balancing via Lyapunov rule-based fuzzy control. *Multibody System Dynamics*, 31(2):147–168, February 2014.

[37] A. N. Gündeş and A. Nanjangud. Low-order simultaneous stabilization of linear bicycle models at different forward speeds. In *2013 American Control Conference*, pages 1840–1845, June 2013.

[38] Jing Yuan, Huan Chen, Fengchi Sun, and Yalou Huang. Trajectory planning and tracking control for autonomous bicycle robot. *Nonlinear Dynamics*, 78(1):421–431, October 2014.

[39] L. Keo and M. Yamakita. Controlling balancer and steering for bicycle stabilization. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4541–4546, October 2009.

[40] Paul Storch. Self-stabilizing training wheels for a bicycle, November 1991.

[41] K. Johannsen. Retractable bicycle training wheels, July 1973.

[42] James Slattery. Deployable training wheels for a bicycle, September 2001.

[43] Jiarui He and Mingguo Zhao. Control System Design of Self-balanced Bicycles by Control Moment Gyroscope. *Lecture Notes in Electrical Engineering*, 338:205–214, January 2015.

[44] K. J. Astrom, R. E. Klein, and A. Lennartsson. Bicycle dynamics and control: adapted bicycles for education and research. *IEEE Control Systems Magazine*, 25(4):26–47, August 2005.

[45] Yasuhito Tanaka and Toshiyuki Murakami. A study on straight-line tracking and posture control in electric bicycle. *IEEE Transactions on Industrial Electronics*, 56(1):159–168, 2009.

[46] K. Chu, J. Kim, K. Jo, and M. Sunwoo. Real-time path planning of autonomous vehicles for unstructured road navigation. *International Journal of Automotive Technology*, 16(4):653–668, August 2015.

[47] Wonteak Lim, Seongjin Lee, Myoungho Sunwoo, and Kichun Jo. Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):613–626, February 2018.

[48] R. Bajaj, S.L. Ranaweera, and D.P. Agrawal. GPS: location-tracking technology. *Computer*, 35(4):92–94, 2002.

[49] Dragan Obradovic, Henning Lenz, and Markus Schupfner. Fusion of Map and Sensor Data in a Modern Car Navigation System. *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 45(1-2):111–122, November 2006.

[50] Kichun Jo, Keonyup Chu, and Myoungho Sunwoo. GPS-bias correction for precise localization of autonomous vehicles. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 636–641, Gold Coast City, Australia, June 2013. IEEE.

[51] Thang Nguyen Van, Trinh Chu Duc, and Tran Duc-Tan. Application of Street Tracking Algorithm in an INS/GPS Integrated Navigation System. *IETE Journal of Research*, 61(3):251–258, May 2015.

[52] Carl Ellis. Parts of a bike diagram: Bicycle anatomy for beginners.

[53] Philipp Sandhaus. bafang-python. https://github.com/philippsandhaus/bafang-python, 2019.

[54] MCC. Bafang bbsxx change settings on the fly (aka 'cop button') and bbsxx communication protocol. https://endless-sphere.com/forums/viewtopic.php?t=94850, 2018.

[55] Arduino AG. Arduino due. https://docs.arduino.cc/hardware/due.

[56] Embedded Systems/Super Loop Architecture - Wikibooks, open books for an open world. https://en.wikibooks.org/wiki/Embedded_Systems/Super_Loop_Architecture.

[57] Nanotec Electronic. CL4-E CANopen/Modbus RTU Technical Manual. https://us.nanotec.com/fileadmin/files/Handbuecher/Motorsteuerungen/CL4-E/fir-v2139/CL4E_CANopen_USB_ModbusRTU_Technical-Manual_V1.4.0.pdf.

[58] T.D. Gillespie, S. A. E. International (Society), and Society of Automotive Engineers. *Fundamentals of Vehicle Dynamics*. Premiere Series Bks. Society of Automotive Engineers, 1992.

[59] Arend Schwab, J Meijaard, and Jim Papadopoulos. A multibody dynamics benchmark on the equations of motion of an uncontrolled bicycle. *ENOC-2005*, pages 511–521, September 2005.

[60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[61] William H. Frey. 2020 Census: Big cities grew and became more diverse, especially among their youth, October 2021.

[62] Alice Tindi and Joy Sales. Return to the blog Social Justice and Ethics: Dilemmas and Opportunities for Psychologists, 2016.

[63] Milos N. Mladenovic and Tristram McPherson. Engineering Social Justice into Traffic Control for Self-Driving Vehicles? *Science and Engineering Ethics*, 22(4):1131–1149, August 2016.

[64] Jason Borenstein, Joseph R. Herkert, and Keith W. Miller. Self-Driving Cars and Engineering Ethics: The Need for a System Level Analysis. *Science and Engineering Ethics*, 25(2):383–398, April 2019.