

Design and Application of Lidar Systems and Electric Bicycle Modeling for Transportation  
Safety

By

Deven Renee Mittman

Submitted to the graduate degree program in the School of Engineering's Mechanical  
Engineering Department and the Graduate Faculty of the University of Kansas in partial  
fulfillment of the requirements for the degree of Master of Science.

---

Chairperson: Dr. Christopher Depcik

---

Dr. Xianglin Li

---

Dr. Lin Liu

---

Dr. James Stiles

Defended: 1/12/2021

The Thesis Committee for Deven Renee Mittman  
certifies that this is the approved version of the following thesis:

Design and Applications of Lidar Systems and Electric Bicycle  
Modeling for Transportation Safety

---

Chairperson: Dr. Christopher Depcik

Date Approved: 1/12/2021

## Abstract

As the importance of reducing dependency on fossil fuels increases, the use of bicycles and electric bicycles (e-bikes) can provide a sustainable and viable alternative for single passenger commuters. The general population's reluctance to transition to bikes and e-bikes is, in part, due to safety concerns and general mobility. However, by designing, modeling, and testing solutions aimed at reducing weaknesses inherent to cycling, perhaps these fears can be lessened. This thesis describes risks currently involved with cycling and supplies potential solutions in the form of blind-spot and road surface monitoring, as well as modeling the body forces and the power consumption of an e-bike in motion.

Chapter 1 introduces the inherent dangers facing bikes and e-bikes when compared to vehicles and discusses methods for improving their safety. Additionally, it highlights the difference between and benefits of preventative safety features for use with cycling. Furthermore, this chapter introduces the use of lidar along with its further use and potential for transportation uses.

In Chapter 2, e-bike blind-spot monitoring is designed and tested using a low-cost two-dimensional lidar system. It describes improvements made from previous versions along with the hardware, software, and capabilities of the design. Testing shows its ability to determine distances to objects while alerting the rider, as well as identifying potential improvements for future systems.

Chapter 3 describes the hardware and software of a hand-held, portable, low-cost three-dimensional (3-D) lidar system designed for road surface monitoring. Thorough testing shows the lidar system's ability to create a 3-D point cloud recreation of a pothole in the road. It also identifies improvements made from a previous version and what augmentations should be made

in future systems. Additionally, this chapter includes a comparison to commercially available lidar systems.

Chapter 4 describes the theoretical equations used to model the motion, body force, power consumption, and battery capacity of an e-bike while in use. Furthermore, by knowing road and weather conditions, e-bike design, combined weight and size of the rider and e-bike, the model can predict battery pack behavior and state of charge.

Chapter 5 details experimental testing of the theoretical model described in Chapter 4. By estimating physical properties from available literature and thoroughly measuring e-bike motion, wind conditions, and route, the model can be compared to the actual battery voltage that was recorded throughout the tested routes. Additional modifications are made to the model to account for discretization of measured data and the state of health of the battery pack to better reflect the collected data.

## Table of Contents

Table of Figures .....	VII
Table of Tables .....	XI
Table of Equations .....	XII
Abbreviations .....	XIV
Acknowledgements.....	XVI
Chapter 1 Introduction.....	1
Chapter 2 Third Generation 2-D Lidar System .....	9
2.1 System Hardware .....	9
2.2 System Software.....	16
2.3 Stationary Data Collection .....	19
2.4 Mobile Data Collection .....	27
2.5 System Diagnosis .....	32
Chapter 3 Second Generation 3-D Lidar System .....	34
3.1 First Generation System.....	34
3.2 Second Generation System Hardware.....	36
3.3 Second Generation System Software .....	41
3.4 Data Collection.....	42
3.5 Performance Optimization .....	47
3.6 System Diagnosis .....	54
3.7 Commercial Systems.....	55
Chapter 4 E-Bike Modeling.....	58
4.1 Introduction .....	58

4.2	Forces Modeling.....	59
4.3	Motor Modeling .....	64
4.4	Battery Modeling.....	66
4.5	Model Results.....	67
Chapter 5	E-bike Testing.....	71
5.1	Introduction .....	71
5.2	Data Collection.....	71
5.3	Data Processing .....	75
5.4	Data Modeling.....	78
5.5	Conclusions .....	88
Appendix A:	2-D lidar system's Arduino microcontroller code .....	91
Appendix B:	Simple Scenario Model Calculations .....	98
References.....		99

## Table of Figures

<b>Figure 2.1</b> Terabee 60 m Evo lidar distance sensor [31]. .....	9
<b>Figure 2.2</b> Arduino Mega 2560 microcontroller [32]. .....	10
<b>Figure 2.3</b> Third-generation 2-D lidar system circuit diagram .....	11
<b>Figure 2.4</b> Bipolar stepper motor used in the 2-D lidar system [35]. .....	12
<b>Figure 2.5</b> Adafruit micro SD card breakout board [37].....	13
<b>Figure 2.6</b> Main housing box model for 2-D lidar system (left) and removable from wall (right) .....	15
<b>Figure 2.7</b> Model of block stand for motor (left) and lidar rangefinder mount (right) .....	15
<b>Figure 2.8</b> Assembled 2-D lidar system, closed (top left), wall removed (top right), and interior (bottom).....	16
<b>Figure 2.9</b> Empty sections of Arduino C++ code. ....	18
<b>Figure 2.10</b> Blind spot LEDs mounted on e-bike handlebars to identify obstacles in left, center, and right lanes. ....	19
<b>Figure 2.11</b> Image of initial 2-D lidar system test area.....	21
<b>Figure 2.12</b> Two sweeps of the 2-D lidar system during initial stationary testing. ....	21
<b>Figure 2.13</b> Only lidar data registered during stationary test is a false-positive of the road divider, visual (left), data model (right).....	22
<b>Figure 2.14</b> Stationary testing with moving vehicles in 30 mph speed limit zone (left) and lit right lane LED with resulting model of snapshot (right).....	23
<b>Figure 2.15</b> Visual of car (above) registered at a maximum speed of 15 mph (below).....	26
<b>Figure 2.16</b> Completed 2-D lidar system mounted on the back of the electric bicycle. ....	28

<b>Figure 2.17</b> Typical visual behind e-bike (left) and sweep data (right) from first mobile test with downward lidar angle.....	28
<b>Figure 2.18</b> Example of successful data collection and blind spot monitoring of stationary vehicles during a mobile test: visual camera (left) and lidar modeling (right).....	29
<b>Figure 2.19</b> Example of successful data collection and blind spot monitoring of moving vehicle during a mobile test: visual camera (left) and lidar modeling (right). ....	30
<b>Figure 2.20</b> Skewed data caused by rapid turning of e-bike during data collection: visual camera (left) and lidar modeling (right). ....	30
<b>Figure 2.21</b> Leaning while turning the e-bike causes a false-positive result: visual camera (left) and lidar modeling (right). ....	31
<b>Figure 3.1</b> Isometric Solidworks CAD model of housing configuration (left) and assembled components (right) [45]. ....	34
<b>Figure 3.2</b> Scanned Formula SAE car and resulting model [45]. ....	35
<b>Figure 3.3</b> 3-D lidar version 2 circuit diagram.....	36
<b>Figure 3.4</b> Adafruit data logger shield [47].....	38
<b>Figure 3.5</b> Arduino Mega proto shield Rev, the second stackable shield used in the 3-D lidar system [48].....	39
<b>Figure 3.6</b> Second version of 3-D lidar system.....	40
<b>Figure 3.7</b> Corner of room scanned for initial 3-D lidar testing (left) and point cloud model of room corner (right).....	42
<b>Figure 3.8</b> Setup of beaker testing .....	43
<b>Figure 3.9</b> Models of empty beaker (left), beaker filled with clean water (center), and beaker filled with dirty water (right). ....	44



<b>Figure 3.10</b> Scanned pothole, with one-foot reference (left) and modeled pothole (right). .....	45
<b>Figure 3.11</b> Second scanned pothole with one-foot reference (left) and modeled pothole (right). .....	46
<b>Figure 3.12</b> Configuration 0: Default isometric view (left), top view (right) of test wall .....	47
<b>Figure 3.13</b> Top view of test wall model with configuration zero/default (a), one/short range (b), two/switching modes, (c), three/maximum range (d), four/high sensitivity (e), and five/low sensitivity (f) .....	49
<b>Figure 3.14</b> Resulting scan outlines with level system (left) and 90° system orientation (right)	51
<b>Figure 3.15</b> First pothole with corrected distance and orientation calculations.....	52
<b>Figure 3.16</b> Second pothole with corrected distance calculation.....	53
<b>Figure 3.17</b> New pothole with one-foot scale (left) and improved model (right).....	54
<b>Figure 4.1</b> Eagle Tree eLogger v4 used on the e-bike to record data for the model.....	60
<b>Figure 4.2</b> BLDC motor efficiency map, scaled to the size of motor used in the e-bike [71]. ....	66
<b>Figure 4.3</b> Resulting battery pack data from modeled scenario.....	70
<b>Figure 5.1</b> BLDC motor efficiency map, scaled to the size of motor used in the e-bike [71]. ....	72
<b>Figure 5.2</b> Two routes for data collection across West Campus at the University of Kansas. ....	75
<b>Figure 5.3</b> Collected raw e-bike speed data and resulting calculated acceleration.....	76
<b>Figure 5.4</b> Raw elevation data and resulting road inclination calculation. ....	76
<b>Figure 5.5</b> Smoothed GPS speed and resulting acceleration compared to original data .....	78
<b>Figure 5.6</b> Smoothed GPS elevation and resulting road grade compared to original data .....	78
<b>Figure 5.7</b> Model of forces acting on the e-bike and resulting tractive force exerted by the e-bike during Route 1.....	79
<b>Figure 5.8</b> Calculated motor power draw throughout Route 1. ....	79

<b>Figure 5.9</b> Model of motor efficiency throughout Route 1.....	80
<b>Figure 5.10</b> Comparison of modeled and measured voltage during Route 1.....	81
<b>Figure 5.11</b> Comparison of modeled and measured battery current draw during Route 1. ....	82
<b>Figure 5.12</b> Comparison of the measured battery voltage, the original model and the model using the adjusted nominal battery pack capacity. ....	84
<b>Figure 5.13</b> Comparison between measured voltage and model including relative nominal capacity and adjusted C-rate. ....	85
<b>Figure 5.14</b> Motor power draw during testing over Route 2. ....	87
<b>Figure 5.15</b> Comparison of measured and modeled battery pack voltage over Route 2. ....	87
<b>Figure 5.16</b> Comparison of measured and modeled current draw from the battery pack over Route 2.....	88

## Table of Tables

<b>Table 2.1</b> Lidar system timing conditions with a 100° sweep angle and optimized software running time of 0.021 s. ....	25
<b>Table 3.1</b> Lidar system measurements at various distances and incident angles.....	50
<b>Table 3.2</b> Comparison of created lidar systems and commercially available lidar systems .....	57
<b>Table 4.1</b> Values for respectively simple modeling scenario.....	68
<b>Table 4.2</b> Calculated model variable remaining constant throughout scenario (Appendix B). ...	69
<b>Table 5.1</b> Physical properties of combined e-bike and female rider .....	72
<b>Table 5.2</b> Weather conditions recorded for e-bike test rides.....	73
<b>Table 5.3</b> Accuracy of measurements from (1) Eagle Tree Micro GPS Expander v4, (2) GPS visualizer [77], (3) Eagle Tree ELogger v4, (4) Wind Sensor Rev. P, and (5) Eagle Tree Temperature Sensor. ....	74
<b>Table 5.4</b> Hausmann-Depcik model calibration for NMC cells.....	82

## Table of Equations

<b>Equation 3.1</b> Lidar measurement, $L$ , to accurate distance, $d$ , in centimeters .....	50
<b>Equation 4.1</b> Newton's Second Law summation of forces acting on the e-bike. ....	59
<b>Equation 4.2</b> Net acceleration force determined with bike and rider mass and measured acceleration. ....	60
<b>Equation 4.3</b> Generic drag force acting on a moving body caused by air resistance. ....	61
<b>Equation 4.4</b> Force of drag acting on vehicle caused by air and wind resistance. ....	61
<b>Equation 4.5</b> Drag force acting on e-bike with wind resistance and adjusted effective drag area. ....	62
<b>Equation 4.6</b> Directivity function to adjust effective drag area based on apparent wind angle [68]. ....	62
<b>Equation 4.7</b> Gradation force accounting for slope of route. ....	63
<b>Equation 4.8</b> Rolling resistance force with a constant, estimated coefficient of rolling resistance. ....	63
<b>Equation 4.9</b> Wheel torque corresponding to the tractive force. ....	64
<b>Equation 4.10</b> Wheel torque to brake torque with motor directly on rear wheel. ....	64
<b>Equation 4.11</b> Motor speed calculated based on linear e-bike speed and tire radius. ....	65
<b>Equation 4.12</b> Brake power output from the electric motor. ....	65
<b>Equation 4.13</b> Motor input power using the power output and motor efficiency. ....	65
<b>Equation 4.14</b> Total calculated power draw. ....	66
<b>Equation 4.15</b> Peukert capacity model expanded to include temperature effects [74]. ....	67
<b>Equation 4.16</b> State of charge calculation using battery pack capacity. ....	67
<b>Equation 5.1</b> Calibration for collected <i>Wind</i> and <i>Temp</i> data by the wind sensor. ....	74

<b>Equation 5.2</b> Hausmann-Depcik battery capacity model [74].....	82
<b>Equation 5.3</b> SOH as a fraction of the nominal capacity.....	83
<b>Equation 5.4</b> Modified SOC for the relative health of the battery pack .....	83
<b>Equation 5.5</b> Adjustment estimation for C-rate to account for SOH .....	85

## Abbreviations

2-D	Two-Dimensions
3-D	Three-Dimensions
$A_f$	Frontal area
BLDC	Brushless, Direct Current
CAD	Computer-Aided Design
$C_D$	Coefficient of drag
DPST	Double pole single throw
E-bike	Electric bicycle
EEPROM	Electrically erasable programmable read-only memory
$F_D$	Drag force
$F_G$	Gradation force
$F_R$	Rolling resistance force
$F_T$	Tractive force
$F_x$	Acceleration force
GPS	Global Positioning System
$I^2C$	Inter-Integrated Circuit
ICSP	In-circuit serial programming
IR	Infrared
KU	University of Kansas
LED	Light-emitting diode
lidar	Light Detection and Ranging
mph	miles per hour

ms	milliseconds
N	Motor speed
NEMA	National Electrical Manufacturers Association
NHTSA	National Highway Traffic Safety Administration
NMC	Lithium nickel manganese cobalt oxide
OpenCV	Open Source Computer Vision
$P_b$	Braking power
PCB	Printed circuit board
$P_m$	Motor power
Q	Capacity
Rpm	revolutions per minute
SCL	Serial clock line
SD	Secure digital
SDA	Serial data line
SOC	State of charge
SOH	State of health
SPST	Single-pole, single-throw
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VDC	Volts Direct Current
$\eta_m$	Motor efficiency
$\tau_b$	Braking torque
$\tau_w$	Wheel torque

## Acknowledgements

Dr. Depcik for supplying thorough NMC voltage discharge data. Alec Jorns for work scaling the motor efficiency map and providing documentation about previous testing procedures and results. My parents and sister for letting me bounce ideas off of them and their continued encouragement and support.



## Chapter 1 Introduction

Methods of transportation can vary for individuals depending on weather, destination, purpose, or other factors. Some might use public transit, personal vehicles, bicycles, or walking as their preferred mode of transportation. Unfortunately, this wide variance of options with disparate speeds results in a complex environment with vehicular collisions accounting for a quarter (24.9%) of all accidental deaths in the United States in 2016 [1]. Understandably, safety is a major concern for most commuters and, while the number of accidents has decreased in the past, there has been a recent rise since 2014 [2].

There are many factors contributing to the popularity of each method of transit. Personal characteristics partially affect travel behavior; for example, aged or childless people are more willing to walk or bike rather than drive [3]. Moreover, external factors contribute to a general utilization of one type of transportation over others. In specific, convenient public transportation leads to a decrease in drivers, pleasant and convenient routes or few available parking places will increase bicycling and walking, and the perception of danger and physical exertion involved with biking or walking increases the numbers of drivers [3, 4]. Furthermore, when environmental conditions (e.g., weather or scenery) are less accommodating for biking or walking, this results in more drivers. However, when those same conditions improve, an increase in bicyclists and pedestrians is far less substantial [4].

In general, the inherent aspects of biking and walking including physical activity and greater safety risks are primary deterrents to using non-vehicular modes of transportation. While the pleasantness and convenience of pedestrian routes can be improved relatively easily by communities, safety and manual effort are mostly left to the individual, particularly for bicycling. Here, cycling requires physical exertion that takes a portion of the rider's attention from their

surroundings and leads to fatigue. As a result, this might lead to poor decisions, subsequently making riders more prone to mistakes and accidents. Hence, reducing the physical effort required while enhancing the awareness of the surroundings should promote bicycle usage for urban environments.

One methodology to make biking easier is to use an electric bicycle (e-bike): i.e., adding an electric motor and battery pack to a pedaled bicycle to assist the rider. E-bikes require less physical work from the rider, making them more accessible to the greater populace. For instance, elderly, disabled, sedentary travelers, and people traveling along steep routes who would normally drive a single passenger vehicle can now ride an e-bike. Moreover, one factor preventing commuters from using bicycles to travel to work is the need to shower after the effort required in riding a bicycle. Here, e-bikes can greatly reduce the need to shower over the same route. Furthermore, they still promote physical exercise and can reduce pollution if used instead of driving or bus riding [5]. In fact, after accounting for both manufacturing and recharging emissions, e-bikes produce significantly less pollutants. When comparing carbon dioxide, hydrocarbon, and fine particulate matter emissions, e-bikes are comparable to busses per person per mile; whereas, they are between two and ten times better than motorcycles, gasoline, diesel, and electric cars. In addition, nitrogen oxide emissions for e-bikes are comparable to the most efficient gasoline cars and far better than all other vehicle options per passenger and per mile [6]. However, even if the physical activity aspect of bicycling to work using e-bikes can be alleviated, there is still a significant safety concern.

Historically, technology has helped to improve safety and save lives in motor vehicle crashes. In a National Highway Traffic Safety Administration (NHTSA) study of the effectiveness of 26 different safety technologies (e.g., seat belts, air bags, shatter resistant

windows, roof crush resistance), the majority of safety features exist to protect the occupants of the vehicle during and after the impact of a crash [7]. While these and other safety measures have been able to decrease the rate of fatalities per 100 million vehicle miles traveled over the past several decades and save 30,000 lives annually, no comparable safety measures have been created to protect cyclists [7-9]. Now, while the number of fatalities and injuries are lower for cyclists than for drivers, cars can transport more passengers at one time and travel a significantly greater distance than bicycles each year. After accounting for the distance each transportation option travels, the difference in safety becomes apparent. For each mile traveled, a cyclist is around ten times more likely to be involved in a fatal accident and fifty times more likely to be injured than a vehicle passenger [10].

Most bike fatalities and injuries occur in urban areas, during rush hours, and while moving at an intersection. This, along with data showing bikes involved in fatal, single-vehicle crashes impact mostly the front of vehicles, implies most bikes are involved in crashes because a car came from behind while both were in motion [11]. In general, the NHTSA says the best way to keep bicyclists safe is to prevent crashes with vehicles altogether [12]. However, for all the advances in vehicle safety and protection, bicycle protection has remained largely the same (i.e., helmet and other padded protective gear). Many vehicle safety features, such as seat belts, airbags, and crumple zones, cannot be applied to bicycles. Of importance, while almost all serious vehicles crashes are a result of human error, vehicle automation, even partially, has the potential to remove this error and reduce the number of crashes [13]. For example, forward collision warning and prevention systems can avert 1.2 million crashes annually [14], but these systems are not feasible for bicycles. In specific, use of an object detection sensor and an intervening braking system that engages faster than a driver's reaction time would introduce a

new danger to the rider if they are not prepared or balanced to come to a sudden stop. However, this distancing sensor feature of collision warning and prevention systems can still be applied to bicycles without potentially adding a new source of hazard to the rider.

A primary safety concern for both vehicles and bicycles is the existence of blind spots. Typically, rear and side-view mirrors help drivers monitor the area behind them and while additional mirrors are suggested to completely eliminate blind spots, watching multiple mirrors will slow drivers' reaction time [15]. Therefore, it is preferable to monitor the area surrounding the vehicle or bicycle via another system. Here, a detection system to alert drivers, visually or audibly, would help improve reaction time while potentially providing more consistent benefit than mirrors alone. In addition, a secondary safety issue plaguing motorists includes the condition of the road. In specific, inadequate road infrastructure is listed as a frequent cause of single-vehicular mishaps, especially roll-over accidents [16, 17]. With the United States infrastructure currently in poor condition, having a detection system monitor road conditions in addition to blind spots would result in a significant opportunity to improve safety [18].

Light detection and ranging (lidar) is one remote sensing method that can facilitate an effective monitoring of both safety concerns. Briefly, lidar works similar to radar systems by using near visible light waves instead of radio waves and can map the surrounding environment in three-dimensions (3-D) [19]. By emitting a pulse of light and measuring the time until the reflected beam is detected in the receiver, the lidar sensor can map the distance to an object with a high accuracy. By knowing the absolute position of the sensor, the time delay between light pulse and reflected pulse, and the angle the pulse was sent, the lidar system can create detailed maps of objects or surfaces. The data is collected in a point cloud data file, typically a .las file, and then modeled using a computer-aided design (CAD) or geographic information system

program [20]. Current applications for lidar systems generally involve aerial platforms and rely on accurate Global Positioning System (GPS) and inertial measuring units to track where the lidar system is when it collects each data point. For example, airborne lidar systems are used for forest mapping to track growth and model forest fire behaviors, classifying land and environmental types, monitoring changes in coastlines, and charting various other environments for a variety of purposes [20-24].

With respect to the safety issues plaguing motorists, ground-based mobile lidar can identify various road types and identify defects in their respective surfaces [25]. In addition, lidar systems can monitor the environment surrounding roads for potential dangers. In areas where valleys and other steep slopes are adjacent to roads, rail lines, and canals, landslides are detrimental to transportation and infrastructure. Therefore, lidar systems can be used to monitor surface material and identify changes and patterns that preclude rock or landslides [26].

While these applications illustrate lidar's propensity to provide accurate and detailed representations, it is often costly to collect this data while respectively difficult to analyze the resulting point cloud. Here, while commercial lidar systems are highly capable, they might be excessive for numerous vehicle-mounted systems. For instance, a vehicular system does not have to scan wide areas of land at a time, only the immediate vicinity if there is a targeted goal in mind (e.g., road conditions versus automated driving). Hence, designing and testing an inexpensive and small lidar system to identify vehicle proximity and road defects could significantly benefit transportation safety while providing for widespread implementation.

As a result, this work will build upon two previous attempts to construct a low-cost lidar system while designing and building a consumer-friendly e-bike. Both versions of the prior lidar systems used the LIDAR-Lite v3 module made by Garmin (range of 40 m and is accurate to +/-

2.5 cm [27]). The first design was constructed primarily to prove the validity of the system and the second system's design was based on the first but modified to be smaller, lighter, and less expensive. Using an Inter-integrated Circuit (I<sup>2</sup>C) architecture instead of serial connections between the lidar rangefinder and the microcontroller was found to provide faster and more reliable data transfer. The second lidar system was designed more purposefully, with the goals of making it smaller, lightweight, relatively inexpensive, and generating fast, complete, and reliable data. It utilized stackable Adafruit Feather System boards to save space and because they are compatible with Arduino programming. These boards controlled the GPS tracking, lidar rangefinder, data logging, stepper motor, and allowed external battery charging along with a Universal Serial Bus (USB) connection [28]. Additionally, each system was equipped with a Raspberry Pi camera to have a visual record of the area scanned by the lidar. It is a low-resolution camera that saved on processing power and was utilized since high resolution was not needed to identify objects in the path of the lidar. Overall, the majority of the second system's weight comes from the battery pack holding eight AA batteries with a 9-volt direct current (VDC) socket connector.

The second system was tested on its ability to identify vehicles and other objects and their distance from the system while in motion. This required the use of Open Source Computer Vision (OpenCV) software to run while the lidar and video cameras collected data. An image classifier was trained using thousands of samples of vehicles and non-vehicles so the video camera system could correctly identify the difference between a car and other obstacles like bushes or lamp posts. The raw data collected from the lidar rangefinder measured the distance to the detected object and, when combined with the angle the stepper motor when that data sample was collected, was used to model the data in two-dimensional (2-D) space. Next, MatLab was

used to model the data and compare it to measured distance and angle the object is from the lidar sensor. During testing, the lidar system was able to correctly identify the center of vehicles with 82.3% accuracy and its position relative to the system with 96.7% accuracy [28].

However, there were several issues identified with the second lidar system. The first is that the entire system swayed and tilted with the bicycle and, as a result, the Raspberry Pi camera did not identify all vehicles due to limitations in its learning capabilities. In addition, the video camera took a photo at instances in time; however, the lidar rangefinder must sweep through the field of view, during which time the distance to surrounding objects changes. As a result, the data collected from the video camera and lidar rangefinder did not perfectly match. Moreover, there was also a significant issue with the data rate collected by both the Raspberry Pi and Garmin lidar rangefinders. Potentially, any vehicle travelling over 19 m/s (42.5 mph) can pass through the field of view without being caught by the system. Furthermore, the combination of the video camera and lidar rangefinder put a strain on the memory and processing power of the entire system; hence, limiting how often data samples could be taken and processed with enough time to provide warnings to the rider with a safe time to react. Finally, the lidar rangefinder also identified any object within its field of vision, vehicle or not, making identifying cars moving near the bike difficult [28].

As a result, this effort describes the design of a more accessible lidar system and the potential impact it can have for transportation safety. This will include the design and testing of a cost-effective, 2-D lidar system mounted on an e-bike to monitor blind-spots and alert riders when a car is approaching. In addition, this effort describes a 3-D lidar system that can be used to monitor road conditions and identify damage. Furthermore, this thesis will work to improve, monitor, and model a battery pack used to power an electric bicycle. The e-bike used for testing

is the second design made for local and urban commuting. However, little research was done into the reliability of the electric motor and battery pack and the influence they would have on the rider behavior [29]. Therefore, modeling e-bike behavior and range can help improve rider safety and trust in electric bicycles. In specific, modeling the total force of the e-bike at any given moment, in conjunction with a lidar rangefinder, can alert a rider when the necessary stopping distance is farther than the closest obstacle. Additionally, by creating a more detailed estimation of how the route and environment affect the state of charge (SOC) of the battery pack, riders will know when to charge the battery, as well as how the route will affect the power draw. If a rider is planning on a mostly uphill ride, the model can show a larger drop in battery charge than a return trip downhill. This model can also estimate an e-bike's capability and allow commuters to determine if it is a suitable alternative to a vehicle given their own lifestyle. While it may be impossible to prevent all accidents and injuries, increasing detection capabilities while alerting riders to hazardous situations should help commuters and drivers avoid collisions with unseen vehicles, bicycles, pedestrians, and unsafe road conditions.



## 2.1 System Hardware

The third generation 2-D lidar system uses a different lidar rangefinder from the previous versions described in Blankenau, et al. [28]. In specific, this new system uses a Terabee Evo 60 m single point lidar rangefinder (Figure 2.1). Here, the Terabee's most significant improvement over the Garmin Lite v3 lidar rangefinder is its detection range. While the Garmin has a maximum range of 40 m, the Terabee Evo 60 m has a maximum range of 60 m [27, 30, 31]. For the purpose of moving vehicle detection, a farther detection range increases the likelihood of sensing incoming vehicles and gives the rider a longer reaction time. In addition, the Terabee was designed for drone applications; hence, it is smaller, costs marginally less, and weighs half as much as the Garmin lidar version. Furthermore, the Terabee does not require external circuitry to be controlled by an Arduino microcontroller [27, 30]. The Terabee Evo 60m sensor uses the I<sup>2</sup>C/Universal Asynchronous Receiver/Transmitter (UART) backboard to connect to a microcontroller using only two communication wires without extra circuitry [30, 31].

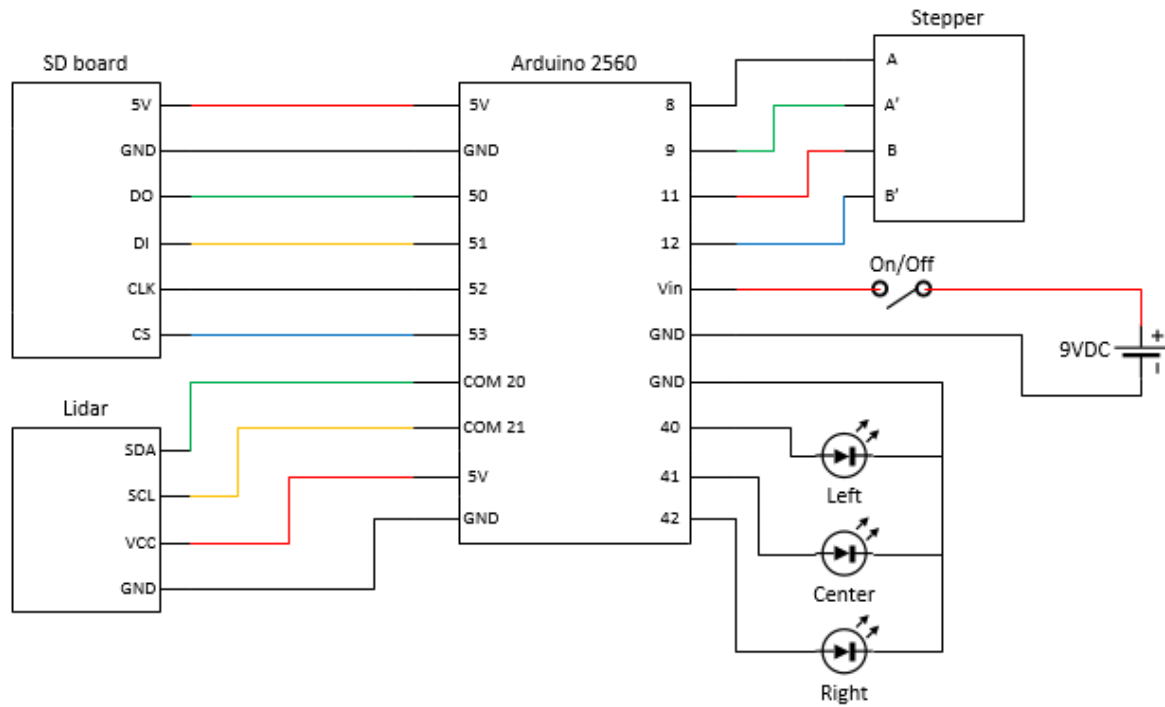


**Figure 2.1** Terabee 60 m Evo lidar distance sensor [31].



**Figure 2.2** Arduino Mega 2560 microcontroller [32].

The Terabee lidar sensor, along with the rest of the system, is connected and controlled by an Arduino Mega 2560 microcontroller (Figure 2.2). The Arduino range of microcontrollers was preferable to other brands due to its extensive online documentation, open-source software, and ease of learnability. While the Mega 2560 is larger than other Arduino microcontrollers, it has 54 digital input/output pins that facilitate communication with the lidar rangefinder, stepper motor, micro Secure Digital (SD) card breakout board, and light-emitting diodes (LEDs) [32]. Additionally, the Mega2560 is powered by a 9 VDC battery while operating and supplying a nominal 5 VDC necessary to power the various aspects of the entire system (Figure 2.3) [33].



**Figure 2.3** Third-generation 2-D lidar system circuit diagram

The microcontroller is connected directly to a QSH2818 stepper motor (Figure 2.4) through four digital output pins without the need for an external motor driver board to complicate the circuit or programming. The bipolar stepper motor is rated for 3.8 VDC to 6.2 VDC and has a National Electrical Manufacturers Association (NEMA) 11 construction. Bipolar stepper motors have only four lead wires compared to unipolar stepper motors that have either five or six; hence, this results in a simpler circuit [34]. This motor's small size and weight are ideal for the goals of the mobile lidar system. Additionally, the maximum torque output of the motor is only 0.07 N·m, yet the lidar rangefinder only weighs 12 grams; therefore, a large torque capability is not a priority [31, 35].

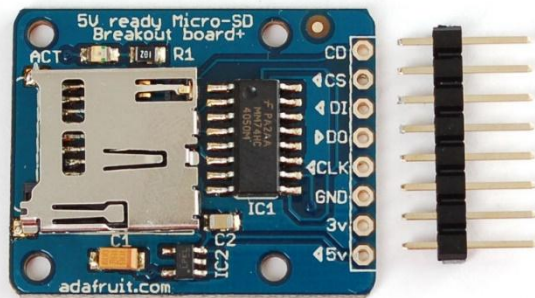


**Figure 2.4** Bipolar stepper motor used in the 2-D lidar system [35].

As a stepper motor turns, it does so in discrete increments that allows the lidar rangefinder to remain at a fixed position during each data sample. The motor chosen has 200 distinct steps per revolution; therefore, each step angle is  $1.8^\circ$  in rotation [35]. While stepper motors with smaller step angles exist, for the purposes of vehicle detection, an average vehicle would have to be just over 60 m away from the lidar system for the change in motor angle to miss the vehicle entirely. Given this information and, as previously mentioned, the lidar rangefinder will have a maximum distancing range of 60 m under ideal conditions, a smaller step angle was determined to be unnecessary [36]. Finally, the motor's flat-sided shaft ensures the lidar rangefinder's mounting will turn with the motor without slipping [35].

The final component powered and controlled by the microcontroller is the Adafruit micro SD breakout board (Figure 2.5). While the Mega 2560 microcontroller can store variables and code script, it cannot store large amounts of data. Typically, it is connected to a computer through the USB port and the computer stores the data. However, to keep the system mobile, the micro SD breakout board can readily store large data files and is connected to the Arduino board through one of the in-circuit serial programming (ICSP) pins [37]. While the micro SD board

runs on a nominal voltage of 3.3 VDC, it also has a 5 VDC pin connected to an onboard fixed-output voltage regulator to lower the voltage and increase the current throughout the board [38]. The micro SD board runs at a relatively high current of 100 mA, twice as much as the Mega 2560 is capable of supplying through the 3.3 VDC power output pin [32, 37]. To make certain the micro SD board will always have the required current, it must be powered by a 5 VDC power output pin by the Arduino microcontroller. In addition, this removable data card allows access to the saved data file without disturbing the rest of the lidar system.

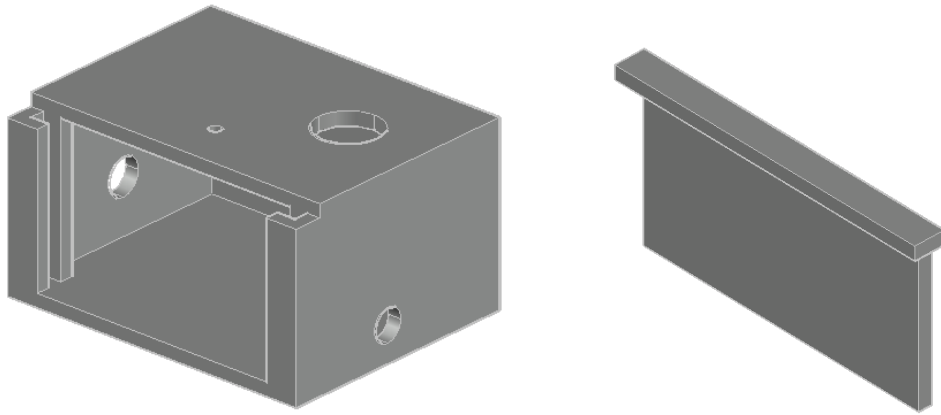


**Figure 2.5** Adafruit micro SD card breakout board [37].

As previously mentioned, the entire 2-D lidar system is powered by a single 9 VDC rechargeable battery. Here, the Arduino microcontroller requires an input voltage between 7 VDC and 12 VDC to adequately supply either 5 VDC at 20 mA or 3.3 VDC at 50 mA to external components [32]. There are three pins available to output 5 VDC to the micro SD card board and the lidar rangefinder. This prevents current from being divided between the components. In addition, digital output pins power the stepper motor and blind spot LEDs were added to alert the rider. In general, Arduino microcontrollers are designed to run their program if they are properly powered. Therefore, a single-pole single-throw (SPST) on/off switch was installed between the 9 VDC battery and the power supply pin on the Mega 2560 microcontroller. When off, the

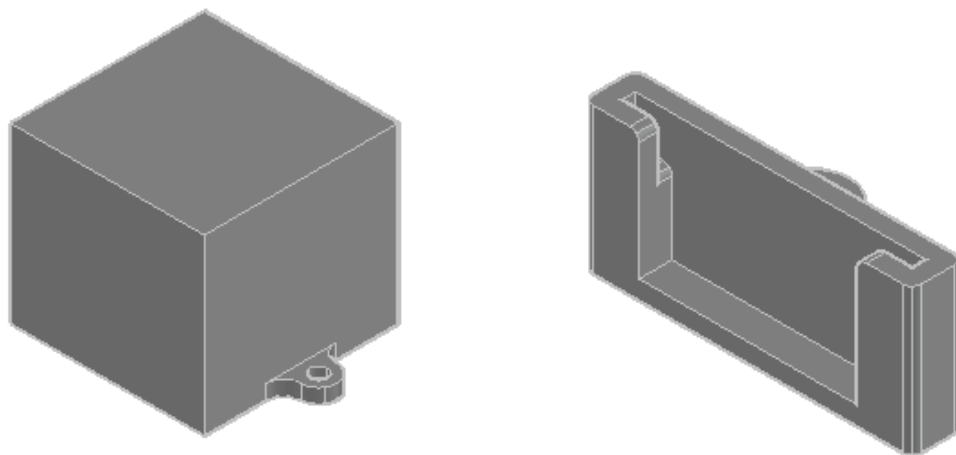
battery's positive line is not connected to the Arduino microcontroller. Once the switch is turned on, the battery is connected, the microcontroller receives adequate power and runs the pre-loaded program from the beginning continuously until the switch is turned off.

Housing and mounting components were 3-D printed using CAD software as illustrated in Figure 2.6. There were several requirements for the housing: it must allow easy access to the circuitry and have holes for the power switch, motor shaft, lidar rangefinder, and blind spot LED wires. Moreover, it must keep out dirt and water that might be encountered while on the back of an e-bike. The largest component, the Arduino Mega 2560 microcontroller, determined the size and shape of the housing box. The microcontroller fits along the back side of the housing wall, facing the removable front wall to readily monitor pin connections. The power switch is mounted to the left side, the motor shaft goes through the top, and LED wires are fed through the housing's right side. Additionally, a slip ring is installed on the top of the housing, connecting the lidar rangefinder to the microcontroller, allowing the rangefinder to turn freely without twisting wires. The front wall is recessed from the rest of the housing to allow a joining slot and can be removed when lifted. This provides access to the microcontroller for reprogramming throughout testing along with access to the battery for charging, as well as the micro SD card for retrieving data.



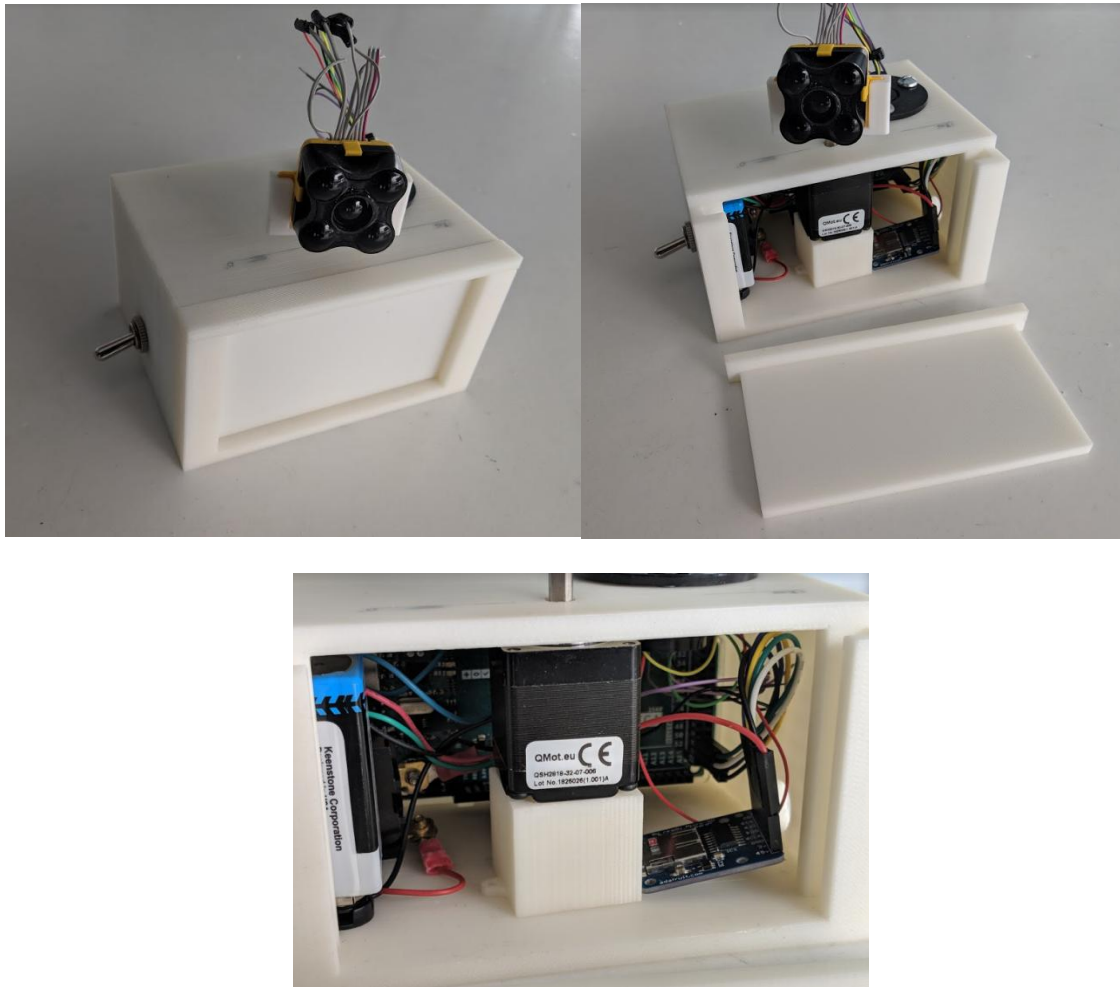
**Figure 2.6** Main housing box model for 2-D lidar system (left) and removable from wall (right)

Due to the housing's height, the stepper motor requires a small block to stand on so that the shaft can reach through the top. The final 3-D component firmly connects the lidar rangefinder to the motor shaft such that it will turn with the motor and not lift off or jostle when the e-bike hits a bump in the road (Figure 2.7).



**Figure 2.7** Model of block stand for motor (left) and lidar rangefinder mount (right)

The housings require 16.5 cubic inches of 3-D printing plastic. The entire system is just over 4 inches tall, 5 in long, and 3.85 in wide. When assembled, the lidar system weighs roughly 1 pound. Additionally, the system costs roughly \$320 excluding 3-D printing costs (Figure 2.8).



**Figure 2.8** Assembled 2-D lidar system, closed (top left), wall removed (top right), and interior (bottom).

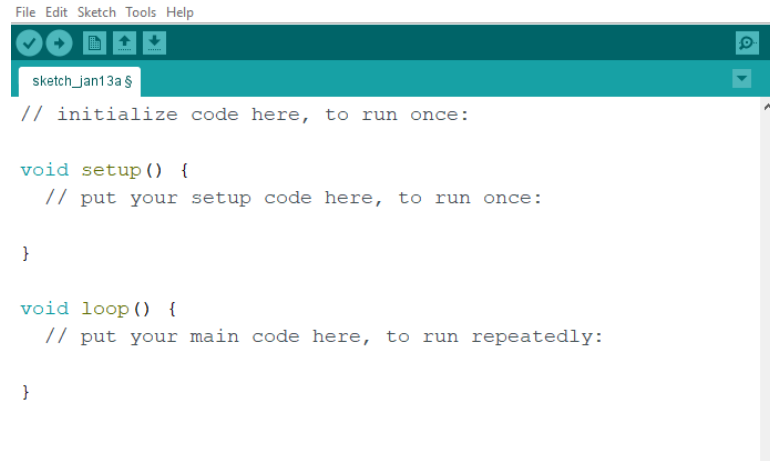
## 2.2 System Software

The Arduino microcontroller uses C++ programming (.ino files) and there are extensive open-source libraries and coding examples available online, as well as wiring connections between the Mega 2560 and each component. In combination, there are collections of coding



functions via libraries (.h files) that serve complementary purposes. For example, the SD library has several functions that work to communicate with an SD card connected to the microcontroller. These functions write data, read data, open data files, and erase data files from SD cards. In general, libraries allow a program to replace dozens of lines of code with one function to accomplish the same task. For the 2-D lidar system, only three libraries were required. The Wire.h library allows for I<sup>2</sup>C communication along Serial Data Line (SDA) and Serial Clock Line (SCL) options used on the Terabee lidar rangefinder [39, 40]. The SD.h library will, among other things, write and save data to the micro SD card [37, 40]. Finally, the Stepper.h library dramatically simplifies commands for the stepper motor [35, 40].

All Arduino program codes have three main sections (Figure 2.9). The first section loads and initializes the libraries and sets up the constants and variables that will be used in the code, as well as their data types. These data types include floating-point numbers (decimal values), integers (whole values), unsigned (value magnitudes), byte storage (any sized object in bytes), and characters (readable letters and words) [41]. The next section of code is the setup and includes items the program only needs to run once upon startup. This section begins the communication between the Arduino and external components in the system, creates data files, and defines pins as output or input signals. The final section of code is the loop that repeats until the code reaches some stall condition or the power supply is disconnected. This section is typically where most of the programming takes place and can involve smaller conditional loops, variable calculations, library functions, and responses to various input signals [40, 41].

A screenshot of the Arduino IDE interface. The menu bar at the top includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for undo, redo, save, and other functions. The main text area shows the following code:

```
sketch_jan13a $
// initialize code here, to run once:

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

**Figure 2.9** Empty sections of Arduino C++ code.

Here, the setup section opens communication over I<sup>2</sup>C to the lidar rangefinder. Next, it identifies the blind spot LED pins and declares them as output signals. Then, it creates, sets up, and saves the data file onto the SD card. Finally, the setup section moves the stepper motor into its starting position. The loop section begins by collecting the line-of-sight distance reading from the lidar rangefinder and converts that information, along with the motor position, to Cartesian distances [39]. Next, the time, sweep count, motor angle, line-of-sight distance, and blind spot LEDs statuses are saved to a text file on the micro SD card. Then, the program compares the  $x$  and  $y$  distances to pre-set conditions to determine if there is a vehicle approaching the e-bike in that direction. These conditions are as follows: (1) was an object detected by the lidar rangefinder, (2) it is closer than 30 meters to the e-bike, and (3) is it close enough to the previous data point to be an incoming vehicle. If these conditions are all met, the program will turn on the LED that corresponds to the lane the data indicates, either the right, center, or left lane (Figure 2.10). The LED will remain on until the rangefinder returns to that point in space and no data fitting a vehicle's criteria occurs. After the LEDs are turned on or off accordingly, the motor turns one step either clockwise or counterclockwise. The motor sweeps an area of roughly 100°

starting at 40° from perpendicular to the direction of the e-bike. At this point, the program loops back to take another data point from the lidar rangefinder and repeats the process until the power switch is turned off (Appendix A).



**Figure 2.10** Blind spot LEDs mounted on e-bike handlebars to identify obstacles in left, center, and right lanes.

### 2.3 Stationary Data Collection

While the lidar system was designed for mobile use, it is simpler to fix bugs and make functional changes before installing the system on the e-bike. Here, several safety conditions were identified as requirements and measurements of success at the start of testing. A typical reaction time of 2 seconds was determined to be the minimum time needed for a bicycle rider to react to an upcoming vehicle [42]. Assuming, when in motion, upcoming vehicles are moving 20 miles per hour (mph) faster than the electric bicycle. This is a reasonable assumption for urban and suburban areas as it is unlikely an e-bike would ride along faster roads, such as highways and freeways. Given the reaction time and speed difference, the critical distance from the e-bike

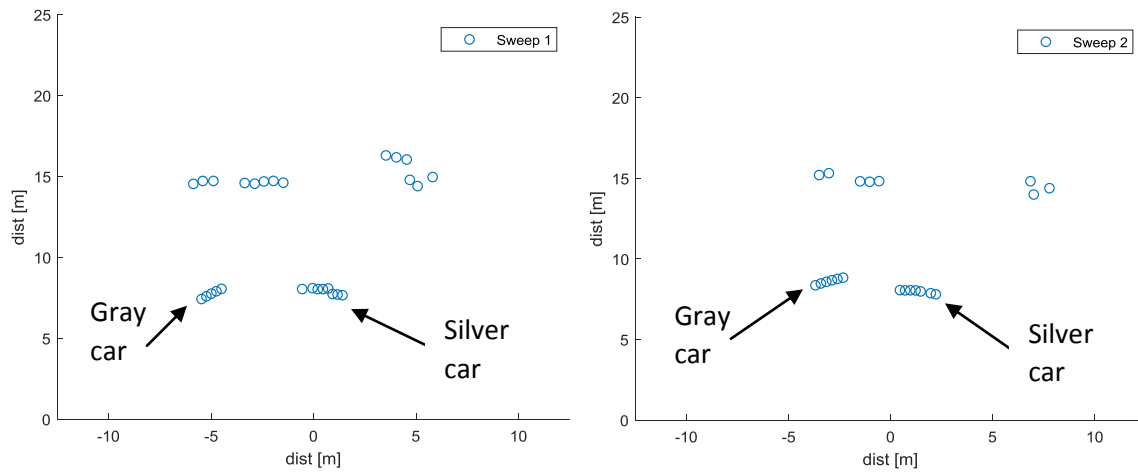
is 58.7 ft (17.9 m). This is the minimum distance behind the e-bike the lidar system must identify a vehicle to signal the rider with enough time to react safely.

A second system requirement is to sweep three lanes behind the bicycle fast enough so a vehicle moving 20 mph faster than the e-bike does not have time to pass the bicycle before the system can identify it. To do so, the lidar rangefinder must sweep from the starting angle through the sweep area of  $100^\circ$  and back within the amount of time it would take a vehicle to drive through the critical distance of 58.7 ft and pass the bike. This results in a minimum motor speed of 16.667 revolutions per minute (rpm) or 1.745 radians per second (rad/s).

The final criterion for success is to distinguish approaching vehicles from stationary or non-vehicle obstacles. Due to the wide variety of vehicle sizes and potential varying speeds between them and the lidar system, this is a more difficult criterion to quantify. Using the minimum step angle possible and the minimum calculated motor speed, the time between each data sample is 0.018 seconds. Assuming an average vehicle speed 20 mph faster than the e-bike and a step angle of  $1.8^\circ$ , the lidar system will theoretically collect between three and six data points per vehicle depending on vehicle size [43, 44]. To account for a vehicle closing the distance to the lidar system at a maximum of 30 mph faster than the lidar system if travelling, a point would be at most 2 ft closer than the previously collected data point 0.018 seconds before. If the lidar rangefinder is pointed at the side of the vehicle and turning opposite to the direction of the vehicle's motion, the second data point would be at most roughly 15 ft farther away from the lidar system than the previous data point.



**Figure 2.11** Image of initial 2-D lidar system test area.



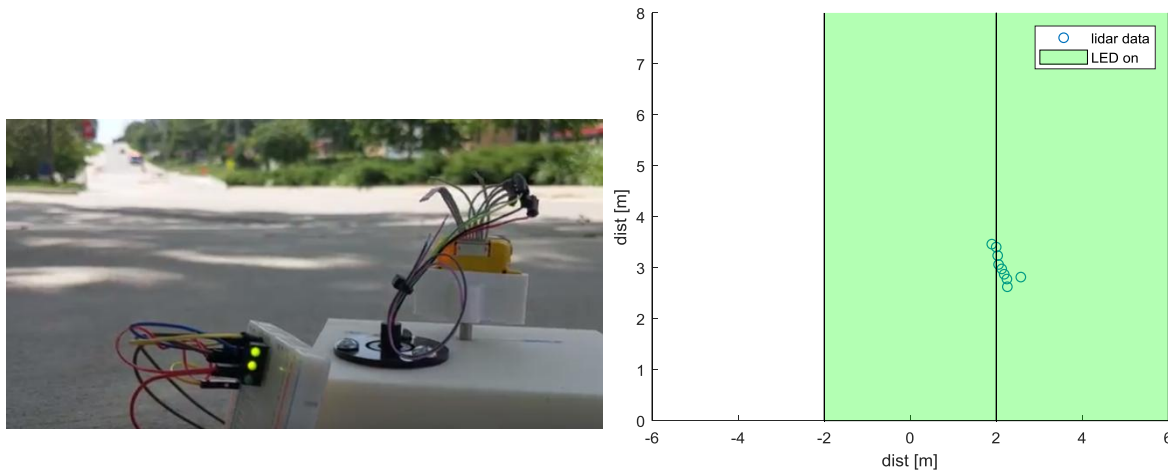
**Figure 2.12** Two sweeps of the 2-D lidar system during initial stationary testing.

After determining the appropriate criteria, the initial stationary tests had the sole purpose of verifying the functionality and accuracy of the lidar system. While held still at roughly 2 feet above the ground, the lidar system was aimed at static cars in a parking lot (Figure 2.11).

Overall, the system was successfully able to map the area accurately and showed two cars were

in front of the wall of the building (Figure 2.12). However, some inaccuracies can be noted in the model recreation of the parking lot. Primarily the data points vary slightly between each sweep of the lidar system. Additionally, the model has a curve to the data and has difficulty showing the difference between the side and rear of the car on the left.

The second stationary testing effort was largely unsuccessful. While on the sidewalk, the lidar system was pointed toward oncoming traffic with a speed limit of 30 mph (Figure 2.13). It is noteworthy that most vehicles slowed as they neared the system possibly out of curiosity or safety concerns. Despite the potentially lower vehicle speed, the lidar system almost never collected data points of these vehicles. The time stamp for each data sample showed the lidar system took 1.9 seconds to turn 100° when it should take a maximum of 1 second. Additionally, the blind spot LEDs cannot accurately distinguish between moving vehicles and empty space.

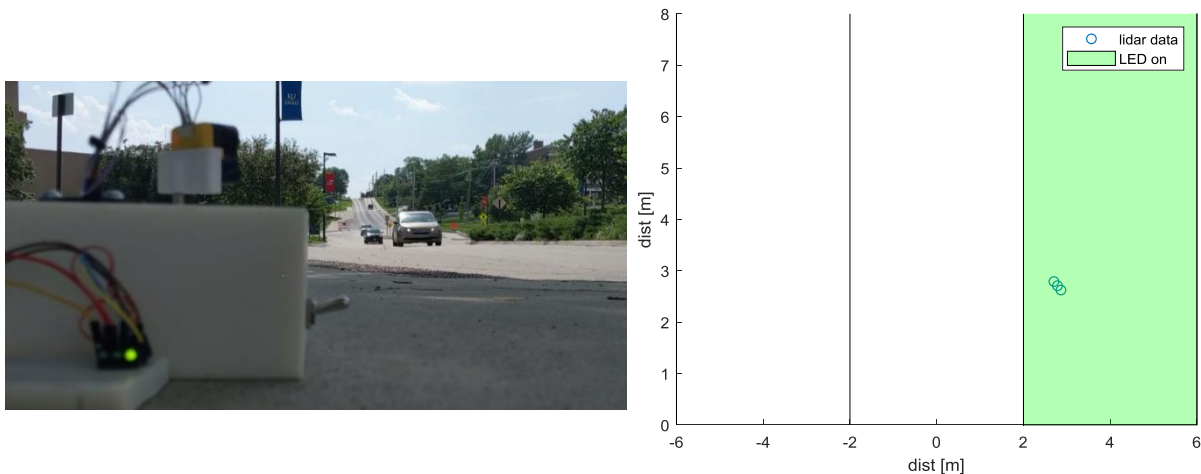


**Figure 2.13** Only lidar data registered during stationary test is a false-positive of the road divider, visual (left), data model (right).

The microcontroller's coding was adjusted to address the flaw in the blind spot LEDs. It was discovered that the center LED would always remain on due to the lidar rangefinder signal. If the lidar rangefinder detects no object, the signal received is a measurement of 1.0. This is read

by the microcontroller as an object one meter away from the system, which results in a permanent object in the center lane. By filtering these data points, the center LED can turn on and off as actual objects enter the view. Furthermore, the stepper motor speed was specified to be 16.667 rpm to sweep 100° in one second.

With these coding changes, the next stationary test had limited success with the lidar system turned to directly face oncoming traffic (Figure 2.14). Out of six passing cars, only one turned on the correct LED. Additionally, the LED remained on for the next three passing vehicles despite the lidar rangefinder collecting no data. Furthermore, there are instances when the lidar rangefinder detected an object which seemingly should trigger an LED but did not.



**Figure 2.14** Stationary testing with moving vehicles in 30 mph speed limit zone (left) and lit right lane LED with resulting model of snapshot (right).

Two potential issues are mostly likely to blame for the performance in the stationary test. The first is the elevation of the lidar system. During the tests, the system was resting on the sidewalk and might only have interacted with the wheels of the vehicles and not their bumpers.

The second issue is that the sweep time of the lidar system was still too slow and several vehicles were able to pass through the field of view before the lidar rangefinder turned in their direction.

The slow rotational speed is affected by the time the system takes to save data to the SD card between each data sample. On average, one line of code running the lidar system takes 0.3 milliseconds (ms), but the singular line of code that saves data onto the micro SD card takes 1.2 ms. Despite best efforts, the code cannot run any faster without risking data corruption. As the lidar system can be turned off at any moment, the system must save each data point as it is collected or risk corrupting the entire data set. Therefore, the microcontroller must save each data point before turning the stepper motor and collecting the next data point. However, the time required for the single line of code to save the data file is four times the amount of time as other lines of code. Additionally, the amount of time required to fully run a loop of the lidar system's code limits the speed of stepper motor rotation. The optimized code will always take 0.021 seconds to run between each data point collected resulting in the lidar system operating slower than the programmed stepper motor speed.

To decrease the time between each data point collected, the motor speed was increased from 16.667 rpm to the maximum usable speed of 50 rpm. This increased motor speed decreased the time between each data point from 2.1667 s to 1.5 s. However, due to the small turning increment, this speed increase is unable to meet the 1 s sweep time requirement (Table 2.1). Overall, the most effective way to decrease the time taken for the lidar system to sweep  $100^\circ$  is to increase the step angle between each data sample. By doubling the steps between each data point from  $1.8^\circ$  to  $3.6^\circ$ , the amount of data collected is halved and the lidar system does not spend as much time saving data; thereby, meeting the previous overall 16.667 rpm threshold. The system, in theory, will always detect a vehicle travelling under 20 mph.



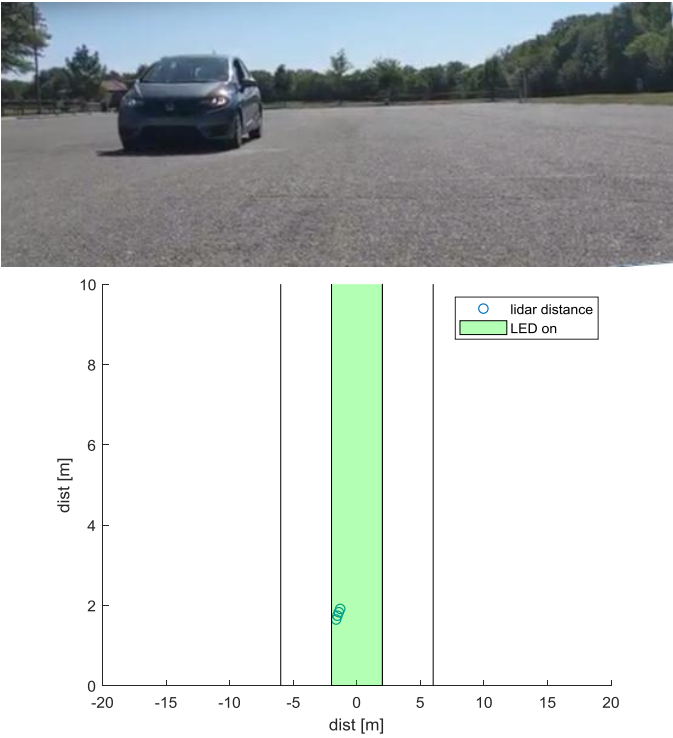
Step Angle and Motor Speed	Motor Turning Time per Step Angle [s]	Data Collection Time [s]	Total Sweep Time [s]
1.8° and 16.667 rpm	0.018	0.039	2.1667
1.8° and 50 rpm	0.006	0.021	1.5000
3.6° and 50 rpm	0.012	0.033	0.9167

**Table 2.1** Lidar system timing conditions with a 100° sweep angle and optimized software running time of 0.021 s.

Subsequently, the final stationary testing effort was more focused. This involved driving one car at 10, 15, 18, and 20 mph towards the lidar system positioned 1.5 ft above the ground on a sunny day. Here, one must take a step back and review the operating principles of lidar. In general, lidar operates via the same fundamentals as radar. A signal is emitted, bounces off a target object, and then is received by the system. The distance to the target is determined from the time delay between emitting and receiving the signal. However, due to the nature of lidar technology, these detection ranges are readily affected by external conditions like infrared (IR) lighting and reflectivity of target's surface material [36]. Furthermore, lidar rangefinders operate using IR light to bounce off a target object and, as a result, any ambient IR light, typically from sunlight, can interfere with these data. Consequently, using lidar systems on sunny days can dramatically reduce the accuracy and range of the sensor [36].

As a result, due to sunny weather conditions during the final stationary tests, the lidar detection distance was reduced and more prone to vehicle detection error. In specific, the maximum vehicle approach speed registered was 15 mph and it was only represented by four data points in the lidar system (Figure 2.15). At higher vehicle speeds, the car passed through the

shortened range of detection faster than the lidar system was able to rotate. Therefore, to capture vehicles moving at higher speeds, the lidar rangefinder would have to sweep the area faster or operate under more favorable weather conditions to extend the detection range and increase the time a vehicle would be noticeable. Unfortunately, the only way to augment the lidar rangefinder speed without altering hardware or electronics is to again to increase the step angle and lose data density. This would not be beneficial since having data points wider apart would increase the likelihood of missing a passing vehicle and/or it would have too few data points for the system to recognize a vehicle. Furthermore, the weather conditions are outside the possibility of control and as such, the lidar system must be able to identify vehicles in most every situation. This is particularly true for weather conditions favorable for bicycle riding; i.e., bright and sunny days.



**Figure 2.15** Visual of car (above) registered at a maximum speed of 15 mph (below).

To balance the speed of rotation of the lidar rangefinder and the density of data points collected, the motor speed was kept at its maximum speed of 50 rpm and the step angle was set at  $3.6^\circ$ . While faster stepper motors would decrease the time turning the motor, the code processing time is the limiting factor for the lidar system's sweep time. Additionally, turning one  $1.8^\circ$  step between data points will not meet the previously calculated system rotation speed due to code processing. Finally, any step greater than  $3.6^\circ$  risks too much space between data points for a vehicle to be missed or not register enough data points to be recognized as a potential vehicle.

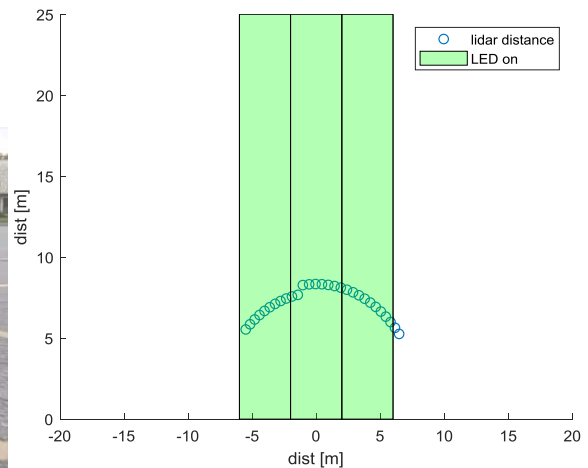
#### 2.4 Mobile Data Collection

With the lidar system operating at the best of its capabilities, it was mounted onto the back of an electric bicycle (Figure 2.16) designed and built by previous students at the University of Kansas (KU) [29]. This e-bike was also used to test the prior vehicle detection lidar system [28]. A large bracket was installed onto the e-bike to hold the lidar system at a suitable height above the ground to better reflect the signal off the front bumper of the car. In specific, the front bumper has a perpendicular angle of incidence to the lidar signal and is a more reliable part of the vehicle to detect. While the windshield offers a larger target, they are slanted and made of glass which offers poor reflection capabilities.



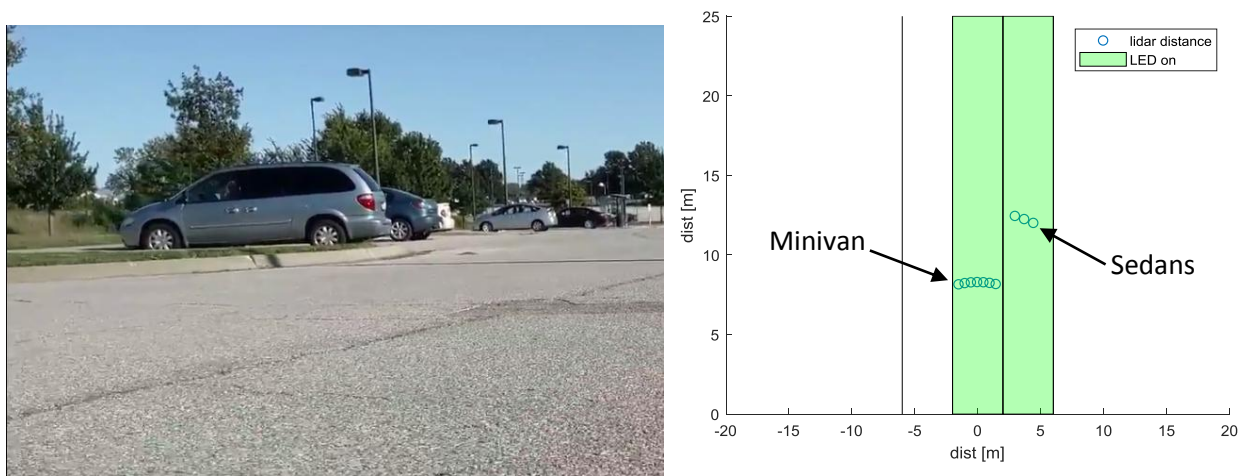
**Figure 2.16** Completed 2-D lidar system mounted on the back of the electric bicycle.

The lidar system was installed on the bracket of the e-bike and had blind spot monitoring LEDs connected to the Arduino microcontroller through a hole in the side of the 3-D printed housing, subsequently attached to the front of the e-bike at the handlebars. The LED states are recorded in the data .txt file along with the lidar distance measurements and motor sweep count. When a blind spot LED turns on, it is modeled with the lidar data by showing that lane in green.



**Figure 2.17** Typical visual behind e-bike (left) and sweep data (right) from first mobile test with downward lidar angle.

To protect the lidar system from the motions and jostling of the e-bike (found in the prior effort to impact the accuracy of the system), a block of insulating foam was attached to the shelf of the bracket under the system. However, after the first mobile test, the lidar system had a slight downward angle which affected the results. As a result, the system would often receive signals reflected from the ground roughly 7 m (20 ft) behind the e-bike (Figure 2.17). To correct the angle of the lidar system, the insulating foam was carved at an angle so lidar system remained parallel to the road surface.

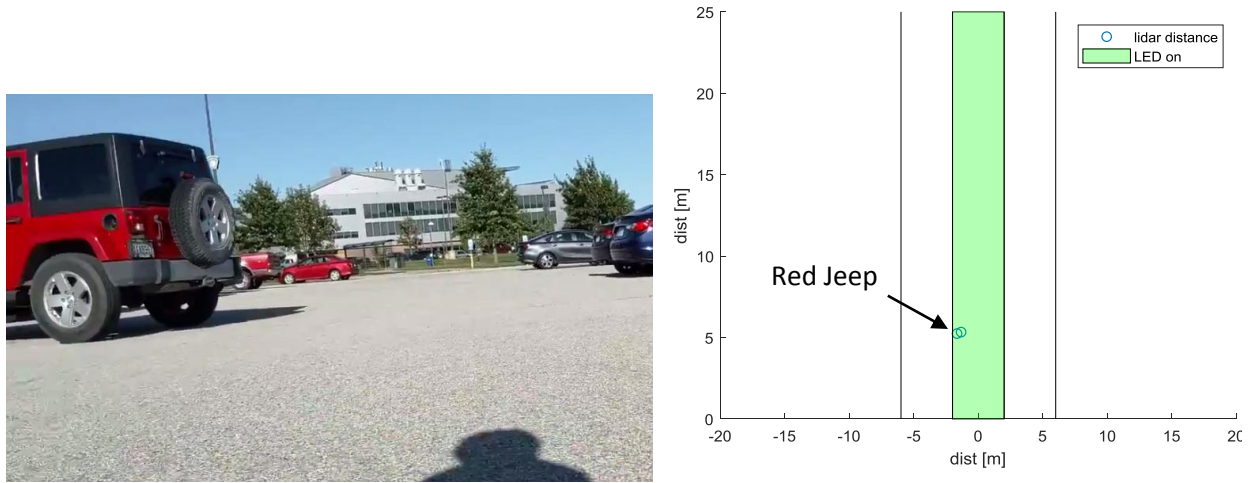


**Figure 2.18** Example of successful data collection and blind spot monitoring of stationary vehicles during a mobile test: visual camera (left) and lidar modeling (right).

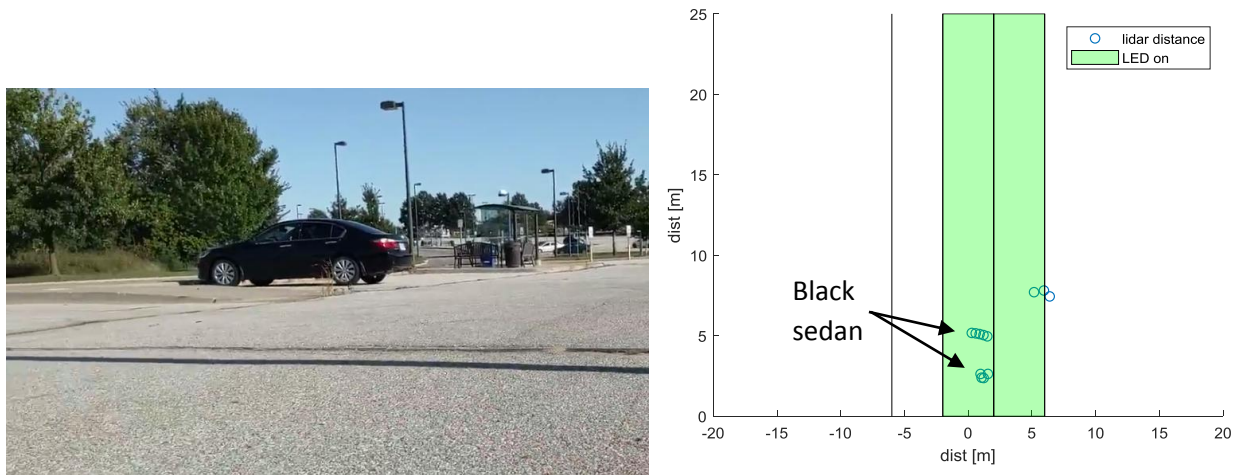
After fixing the angle of the lidar system, a mobile test occurred that involved riding the e-bike around a parking lot next to parked cars, slow-moving cars, and bushes at the edge of the pavement. Throughout this test, the lidar system was powered on and collected lidar data, motor angles, and the states of the blind spot LEDs. In addition, a video camera was set up to record the area behind the e-bike to match the lidar data to specific objects.

This mobile testing demonstrated several promising results. The lidar system was able to detect stationary (Figure 2.18) and slow-moving vehicles (Figure 2.19) in a parking lot at

accurate distances and positions relative to the e-bike. However, moving vehicles were more likely to be missed as they typically do not register as many data points using the lidar sensor.



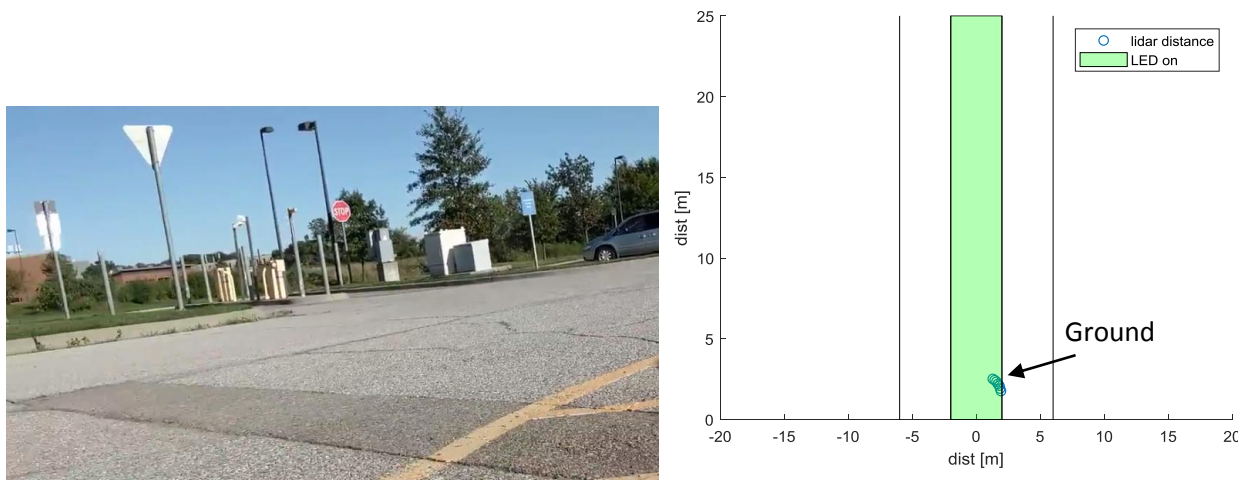
**Figure 2.19** Example of successful data collection and blind spot monitoring of moving vehicle during a mobile test: visual camera (left) and lidar modeling (right).



**Figure 2.20** Skewed data caused by rapid turning of e-bike during data collection: visual camera (left) and lidar modeling (right).

Unfortunately, the lidar system is not able to account for every condition. During mobile testing, the e-bike turns slightly while in motion. Therefore, it is not always oriented in the same

direction throughout a single sweep. This can lead to data points appearing behind other data points along with other skewed data results (Figure 2.20). Additionally, when the e-bike tilts on its side while turning, the lidar system is momentarily pointed at the ground on one side and cannot distinguish these data points from actual obstacles or vehicles (Figure 2.21). However, these false positives occur nearly every time the e-bike turns; thus, making them predictable. Here, adding another criterion to the microcontroller code to ignore lidar data too close to the lidar system would eliminate these false positives but might result in an increased risk to the rider due to close vehicles.



**Figure 2.21** Leaning while turning the e-bike causes a false-positive result: visual camera (left) and lidar modeling (right).

Finally, the lidar system is sensitive to jostling. Sharp vertical motion, caused by a pothole or large crack in the pavement, can cause a momentary loss of power to the system. As a result, the program restarts after the bump and the motor turns as if it is at the start of a sweep. This can cause the lidar system to only scan on one side of the e-bike. Therefore, all mobile testing must be done cautiously and at a slower speed to minimize the impact to the system. The

best potential solution here is to create a more secure connection between the microcontroller and the rest of the lidar system and implement a better shock absorbing system than foam.

## 2.5 System Diagnosis

This third generation of a 2-D lidar system involved several changes from the previous final version in the Blankenau et al. paper. This new system more closely resembles the first-generation system discussed as it is simpler in construction. Arduino microcontrollers are more multi-purpose, easier to learn, and adaptable with circuitry design changes than the Raspberry Pi and Adafruit Feather stackable system used in the second generation lidar system [28]. However, this ease of use and design comes with slower processing speeds. The Raspberry Pi Model B microcontroller operates at 1 GHz compared to the Arduino Mega 2560 at 16 MHz. It would improve performance of the lidar system to return to the faster Raspberry Pi and Adafruit Feather circuitry to dramatically reduce the time required between each data point collection.

A Terabee lidar rangefinder is used on the new lidar system because it has a greater range of 60 m at a comparable size and weight. This lidar rangefinder also does not require an external capacitor between the connection to the microcontroller [27, 31]. Additionally, the new lidar system does not include a visual camera as the previous lidar system. This camera, while important for a visual record of testing also uses OpenCV vehicle recognition software to visually distinguish between cars and other objects. This software requires a large database of known vehicles and non-vehicle images to compare to data. This is not necessary for another lidar system as it adds more circuitry and slows data processing. Therefore, the new lidar system does not include an integrated visual camera system.



The majority of the previous lidar system's weight is a result of the battery pack made of eight AA batteries. The new lidar system requires only one 9 VDC battery. Additionally, the new system is more user-friendly. It includes a single on/off switch, and markings on the housing to show the lidar rangefinder's starting position. Furthermore, all circuitry is contained inside the relatively weatherproof 3-D printed housing. The addition of a slip ring allows the lidar rangefinder to turn freely without twisting or pulling on the wires connected to the microcontroller. Finally, this new system includes LEDs and coding to attempt vehicle recognition solely from lidar data.

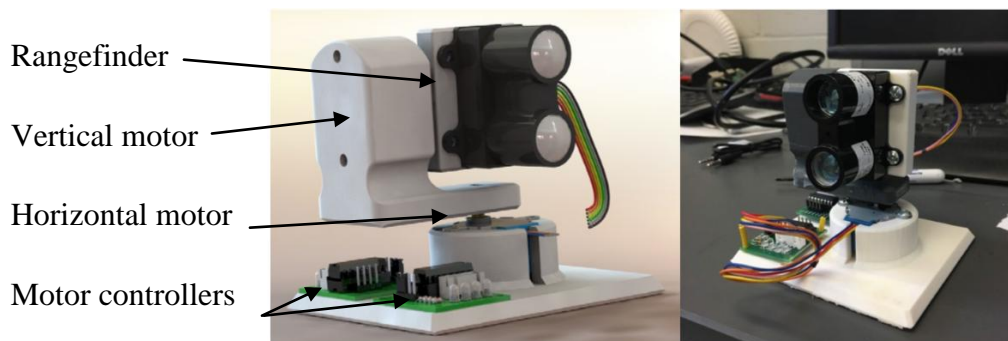
Any future lidar systems based on this or previous systems should include faster microcontrollers. The Raspberry Pi used in Blankenau et al. is a better choice than the Arduino Mega 2560 and the Terabee 60m Evo lidar rangefinder has a better range and simpler operation than the Garmin LIDAR-Lite v3. Additionally, the compact structure and direct soldering connections on the Blankenau et al. lidar system should reduce potential wiring issues if the system is jostled. Furthermore, a 3-D accelerometer could add another source of information that could tell when the e-bike is turning or leaning, and potentially adjust for skewed data while ignoring false-positive results from the ground.

Lidar systems will always suffer from temperamental operation. Weather conditions will continue to affect ranging distances, and surface reflectivity, opacity, and angle of incidence can alter a lidar reading. Additionally, true vehicle identification may never be possible using only 2-D lidar as a grouping of data points roughly indicates an object's width. The different variables affecting how a vehicle or other object approaches a lidar system means it is nearly impossible to account for every situation with a single model. Overall, more work is needed to provide repeatable data suited for various traffic and weather conditions.

### 3.1 First Generation System

An undergraduate team at the KU was prior tasked with designing, building, and testing a small and inexpensive lidar system to scan in 3-D [45]. Here, a portable and accurate 3-D lidar system has numerous potential applications for transportation safety including blind spot monitoring and road surface conditions. The final design of this previous effort included an Arduino Mega 2560 microcontroller, a Garmin Lidar Lite-v3 rangefinder, two stepper motors, two stepper motor controllers, a 5 VDC power supply, and a USB connection to a laptop or desktop computer. Overall, this system could produce detailed point cloud models at a reasonable price but was neither lightweight nor easily portable.

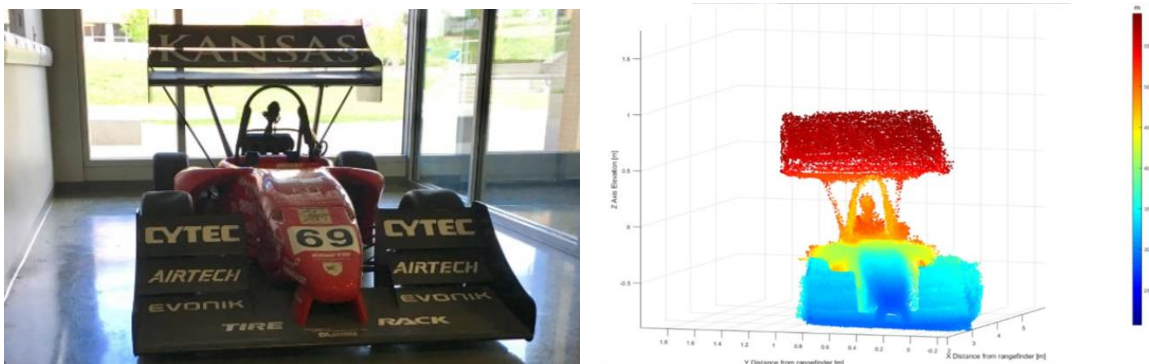
Three 3-D printed plastic housings were designed and made to hold the stepper motor controllers along with the horizontally positioned stepper motor, the vertical stepper, and the lidar rangefinder (Figure 3.1). These housings were designed to keep the lidar rangefinder stationary at a fixed point in space while rotating around that point. This reduced potential error caused by the lidar rangefinder moving and altering the captured distance with respect to its surroundings.



**Figure 3.1** Isometric Solidworks CAD model of housing configuration (left) and assembled components (right) [45].

This 3-D lidar system required several components to work properly. An external 5 VDC power supply, connected to a wall outlet supplied power to the stepper motor controllers, lidar rangefinder, and microcontroller at the same time. The USB connection between the laptop computer and the Arduino microcontroller served to send the data to a separate serial monitor program, as well as supply supplemental power to the microcontroller [45]. Without both the power supply and computer USB connection, the lidar system would not have the required power to function. Since the power supply must be plugged into a wall outlet, the system was usable only in specific locations.

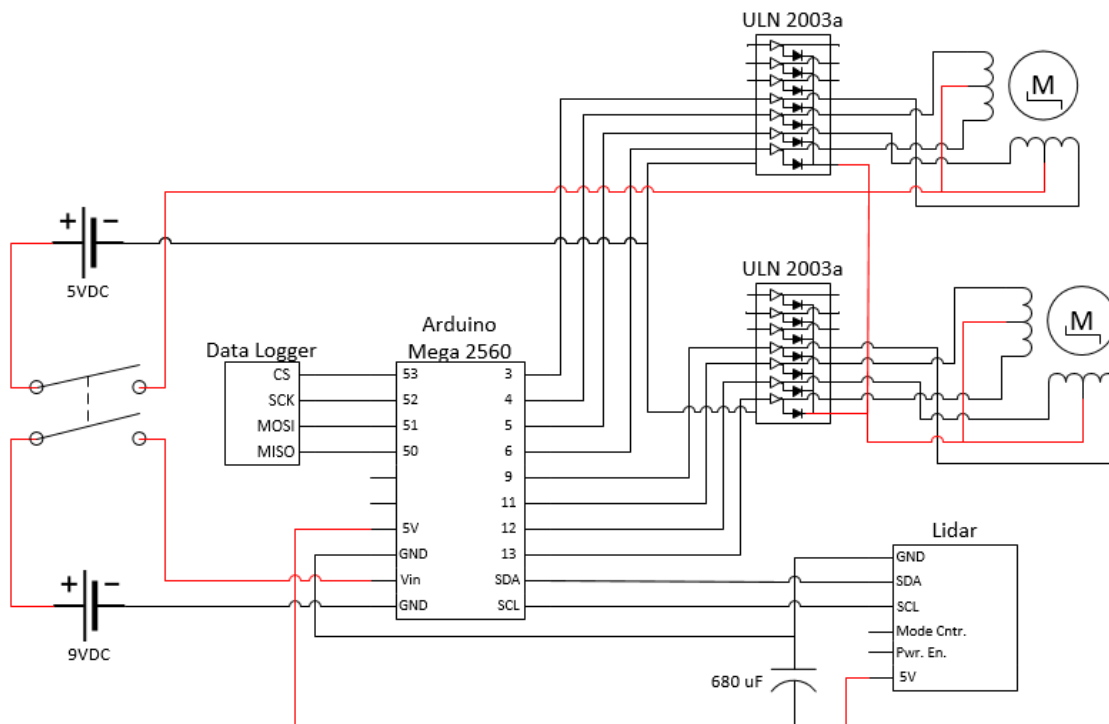
To achieve detailed point cloud models with this system, the stepper motors turned one step ( $0.088^\circ$ ) between data points. Therefore, the resulting point clouds had hundreds of thousands of data points. Subsequently, collected data files required around one to two hours to gather information, depending on the size of the object. The lidar system's data was converted to  $x$ ,  $y$ , and  $z$ -coordinates and modeled using a 3-D scatter plot in MatLab (Figure 3.2). While the lidar system was able to collect accurate data and produce respectively clean models, the system was unsuitable for mobile use [45]. Here, this chapter describes the changes to a subsequent version of this 3-D lidar system with the goal of being self-powered and mobile while having the ability to save data without a serial monitor connection.



**Figure 3.2** Scanned Formula SAE car and resulting model [45].

### 3.2 Second Generation System Hardware

A second version of the 3-D lidar system improves on some of the previous design flaws, decreases the size and weight, and improves mobility. This second version keeps the system circuitry and electrical components as similar as possible while allowing these improvements. The Garmin lidar rangefinder, Arduino microcontroller, motors, and motor housings are the same as the first version to lessen the potential for electrical problems. However, the motor controller circuit boards, capacitor for the lidar rangefinder, and connections made on the breadboard are soldered directly onto stackable protoboard shields. Additionally, the second 3-D lidar system includes a self-contained, battery power supply, data storage, and a power switch (Figure 3.3).

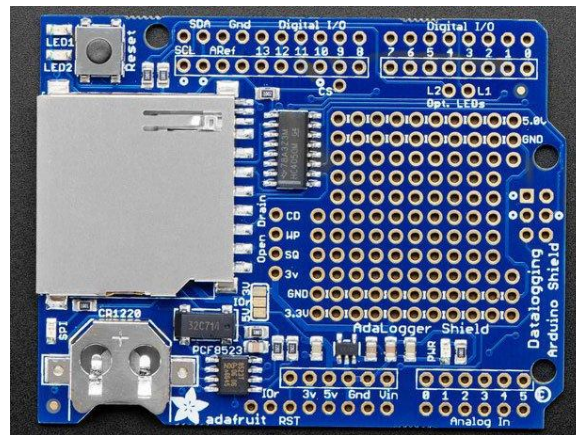


**Figure 3.3** 3-D lidar version 2 circuit diagram.

The major issue with the first system is the necessary connections to both a relatively large power supply connected to a wall outlet and a computer to both supply power and collect data in real-time. The goal here is to make the second 3-D lidar system completely portable and respectively easy to transport and use. Therefore, the power supply must be self-contained in the system. This power is divided between two battery packs located at the bottom of the system. The voltage of the first battery pack is 9 VDC and supplies power to the Arduino microcontroller through the power switch connected to the inlet voltage and ground pins [46]. However, this battery pack does not provide enough power for every component of the lidar system through the microcontroller. Specifically, the two stepper motors require more current than the Arduino itself can supply, which causes a rapid voltage drop and results in the system turning itself off for protection. Therefore, a second 6 VDC battery pack supplies power directly to the stepper motors through the same power switch as before; however, without running current through the microcontroller. It is important to note that the stepper motors are still controlled by the Arduino microcontroller.

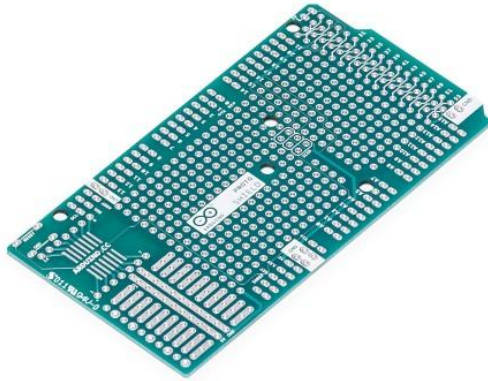
A second issue with the initial 3-D lidar system was its inability to save data onboard the system. The data points were collected through the serial readout on a connected laptop computer and manually saved to a .txt file once data collection was finished. The second 3-D lidar system addresses this issue by including a SD card. This SD card connection is pre-mounted on a data logging shield that has an area for direct soldering of circuitry to the board [47]. The Adafruit data logging shield (Figure 3.4) is the same size as an Arduino Uno microcontroller and is smaller than the Mega 2560 microcontroller used in this system. However, the two boards are still stackable, with the Mega 2560 extending past the end of the data logging shield. This top shield holds the double-pole, single-throw (DPST) power switch and lidar

capacitor, as well as connecting the data storage lines through the ICSP connections to the necessary pins on the Arduino microcontroller. Furthermore, because this shield does not have obstructions to tangle wires as they move, the lidar rangefinder's I<sup>2</sup>C and power lines are connected to the microcontroller through the corresponding pins on this shield.



**Figure 3.4** Adafruit data logger shield [47].

In addition, stackable shields allow the Arduino Mega 2560 microcontroller to expand the number of possible connections and they allow for circuit connections directly on the board without the need for the less permanent breadboard connections used in the prior system. The second shield included is an Arduino Mega Proto Shield Rev3 (Figure 3.5). It is the same size as the Mega 2560 and is a printed circuit board (PCB) [48]. Here, the circuitry needed to power and control the stepper motors and the SD card are soldered onto this shield. To save space, the motor controller boards in the first design were eliminated and recreated with the same transistors on this proto shield with the stepper motor pins connected directly to this shield [49]. Additionally, the ICSP lines from the top data logging shield are connected to the microcontroller through this protoboard shield.



**Figure 3.5** Arduino Mega proto shield Rev, the second stackable shield used in the 3-D lidar system [48].

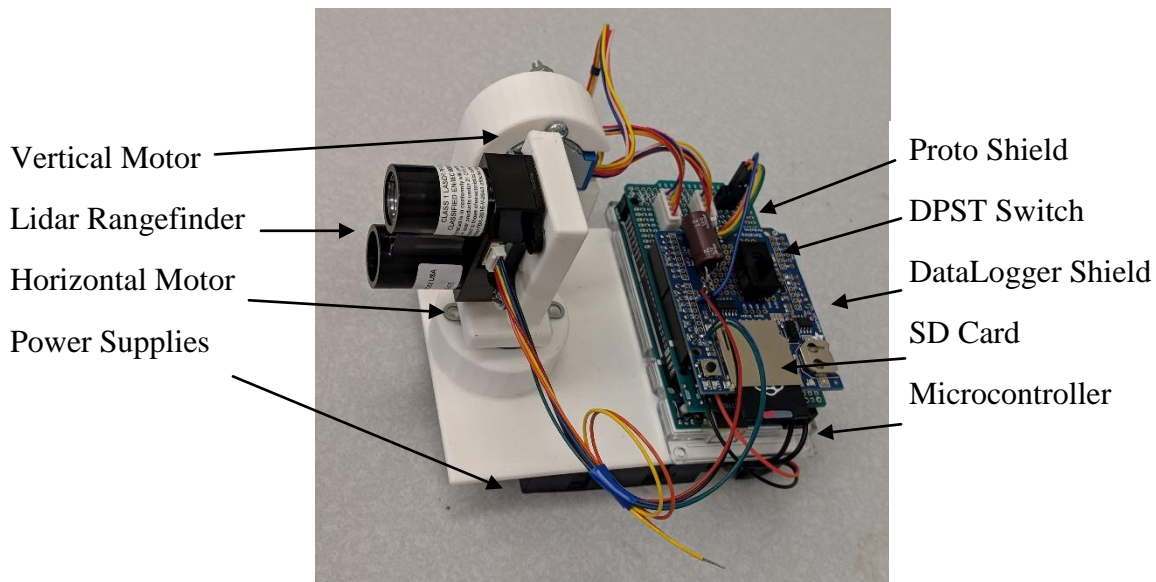
The stepper motors and their corresponding housings are unchanged from the first 3-D lidar system. As before, the horizontal motor turns the vertical motor and its housing, and the vertical motor turns the lidar rangefinder. However, as previously indicated, the motor controller boards were eliminated and replaced with their primary components onto the second protoboard shield in the new 3-D lidar system. Each motor controller board consisted of pin connections to the motor and the Arduino microcontroller, a ULN 2003a Darlington transistor, power supply connections, and indicator LEDs to show the state of the motor at any given time. As the LEDs are not necessary for proper motor function, they are not included in the new system hardware.

The two components from the motor controller boards necessary to include are the Darlington transistors and female pin connectors. The transistors send the control signals from the microcontrollers to the motors while supplying each motor with power. Each female pin connector fits the male connector of the corresponding motor to ensure connectivity [49]. While Darlington transistors are relatively short and readily fit beneath the data logger shield installed

above the protoboard shield, the female motor pin connectors are too tall and are instead mounted in the area not covered by the smaller data logger shield.

As previously mentioned, the Garmin lidar rangefinder is the same version used in the prior 3-D lidar system. Therefore, the correct connections between the rangefinder, 680  $\mu$ F capacitor, and microcontroller pins are already known. The Garmin Lidar Lite v3 rangefinder is connected to through the top data logger shield to allow freedom of motion as the rangefinder rotates during data collection. This rangefinder has several different modes of operation suitable for dissimilar purposes ranging from long and short-distance measurements, high speed and high accuracy, and a general balance of data collection.

Assembled, the new 3-D lidar system is a fraction of the size and weight of the previous system (Figure 3.6). In addition, the new system is entirely self-contained and portable. It does not need to be connected to a power outlet or computer to receive power or save data.



**Figure 3.6** Second version of 3-D lidar system



### 3.3 Second Generation System Software

The program code has undergone several alterations from the first version of the 3-D lidar system. First, the new code for the lidar system includes the Stepper.h library to simplify control of the motors. Without the library, the eight possible input combinations of the motors' pins must be explicitly detailed at the start of the code. In the new version, the library reduces the initialization down to one line of code and performs the same functions that turns the motors as needed. In general, the horizontal motor will turn one step after every data point until it reaches the end of its sweep angle at which point it turns the opposite way. When the horizontal motor changes direction, the vertical motor turns one step upwards. As a result, the lidar rangefinder covers a 3-D space and never collects the same data point twice.

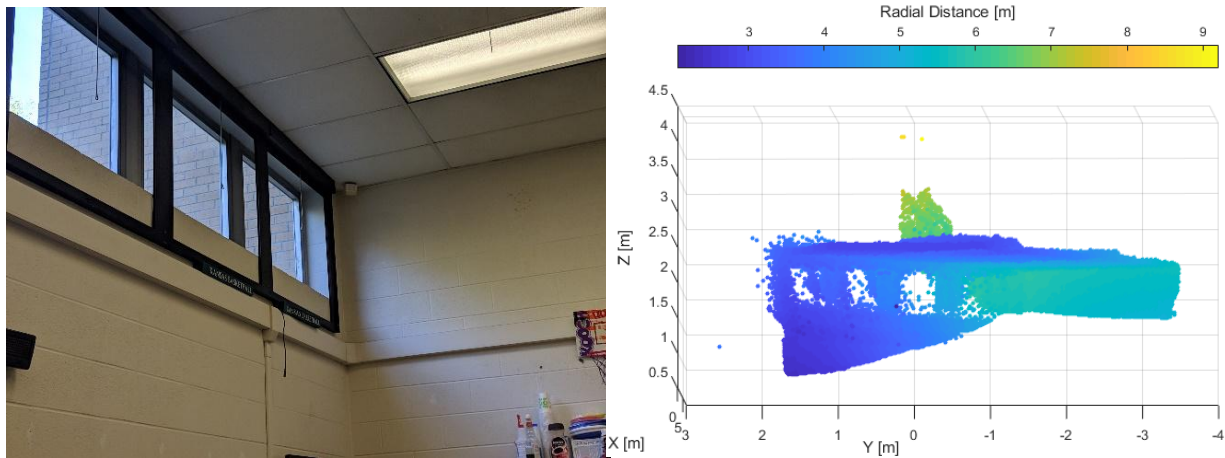
The second major change from the first program is the inclusion of the SD.h library. Since the first 3-D lidar system did not utilize a memory card, there was no need to use this library. As mentioned in the 2-D lidar section, the SD.h library allows the system to communicate with, access information, create, edit, and save data files onto an SD card [40]. The data saved to the SD card is a .txt file consisting of the horizontal motor's angle (azimuth), the vertical motor's angle (elevation), and the lidar distance measurement.

The system is programmed to save as each data point is added to the data file and will end the program once the entire set area has been mapped. The data file is then uploaded to a computer to be modeled using MatLab. This MatLab code converts the azimuth, elevation, and distance values to corresponding Cartesian points, filters out extraneous or flawed data points, and creates a scatter plot of the data.

### 3.4 Data Collection

The initial performance test run with the ungraded 3-D lidar system included modeling a portion of a room. This served to identify potential issues with the system while providing a performance comparison with the previous 3-D system. As with the 2-D lidar system, the lidar data and motor positions are saved to a data file and modeled as a point cloud in MatLab. The initial positions of both motors and the system's orientation must be noted for each test as these factors will affect the accuracy of the computer model.

The portion of the room in the initial test covered an upper corner, windows, and an exterior wall visible through the windows. The 3-D lidar system scanned  $90^\circ$  horizontally and  $45^\circ$  vertically resulting in over 123,000 data points and took nearly an hour. Overall, the resulting model is relatively accurate (Figure 3.7).



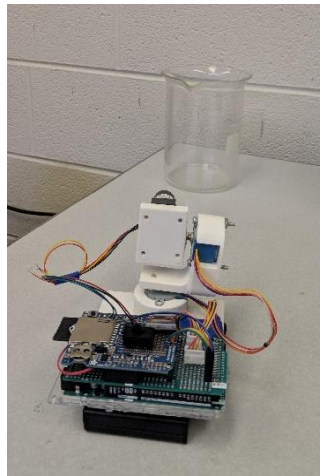
**Figure 3.7** Corner of room scanned for initial 3-D lidar testing (left) and point cloud model of room corner (right).

However, the point cloud model is less accurate around the windows and the recessed lighting in the ceiling. Specifically, the model shows large rectangular recesses in the ceiling

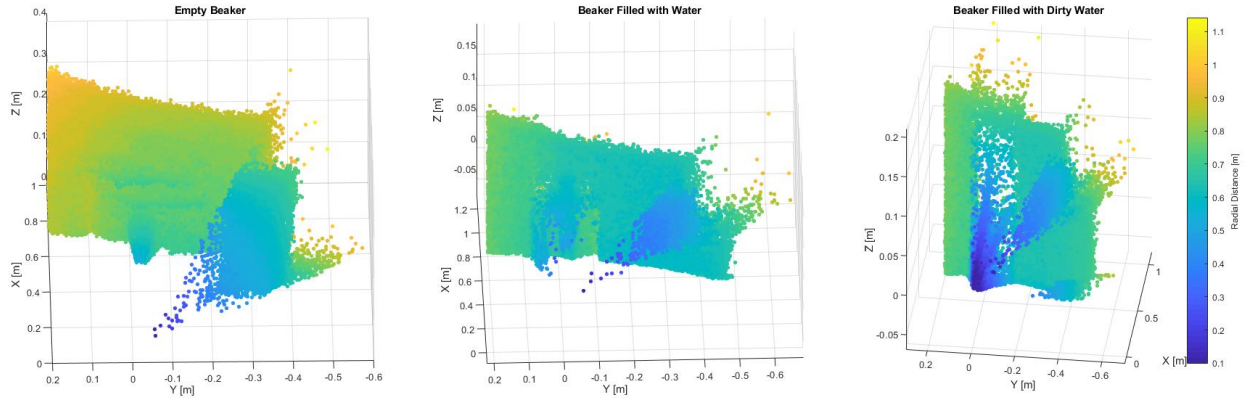
where lighting is located. Here, this could be possibly due to the corrugated texture of the lighting panels or that using lidar to directly scan a source of near infrared light interferes with its distance calculations.

Furthermore, the 3-D lidar system can detect the exterior wall visible through the windows. This is not surprising as infrared light has a wavelength near visible light and will behave similarly. As the wall can be seen through the glass by the human eye, the 3-D lidar system can also detect its presence. As seen, the data collected is not perfect through glass as this material will diffract the signal and skew the output (Figure 3.7).

Secondary testing was then accomplished to better understand how the 3-D lidar system behaves modeling different materials. The most common non-opaque material it might encounter outside is water. As such, the test included modeling an empty beaker, a beaker filled with clean water, and a beaker filled with dirty water. The lidar rangefinder was set on a table facing the beakers (Figure 3.8) and the results are shown in Figure 3.9.



**Figure 3.8** Setup of beaker testing



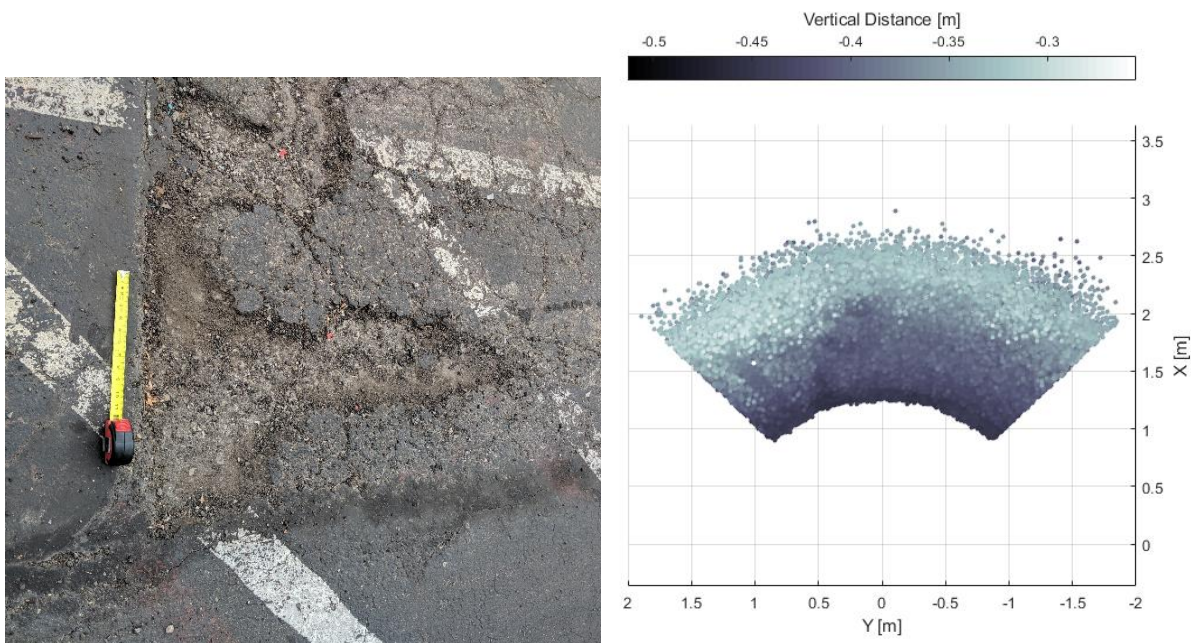
**Figure 3.9** Models of empty beaker (left), beaker filled with clean water (center), and beaker filled with dirty water (right).

As expected, the empty beaker did not interact with the lidar system as an opaque object would. Instead, both the empty and clean water beakers reflected the lidar signal at the curved edges of the beaker, where the lidar signal would pass through the most amount of solid glass. At a more perpendicular angle, the glass and clean water allowed the signal to pass straight through and the system only detected the wall behind the beaker. The beaker filled with dirty water was able to interact with the lidar signal somewhat, but still did not result in a model showing a beaker shape.

Since a potential use for this 3-D lidar system is to model road conditions and map potholes in the surface. The next 3-D lidar system test involved capturing potholes while stationary. Due to the housing design, the lidar rangefinder is unable to point at a steep downward angle. Therefore, to map a pothole, the lidar system must either be placed further away from the pothole, or the system can be turned on its side such that the housing no longer inhibits the motion of the lidar rangefinder. Therefore, to increase data point density and limit the

possibility of external interference, the lidar system was situated near the potholes and oriented sideways.

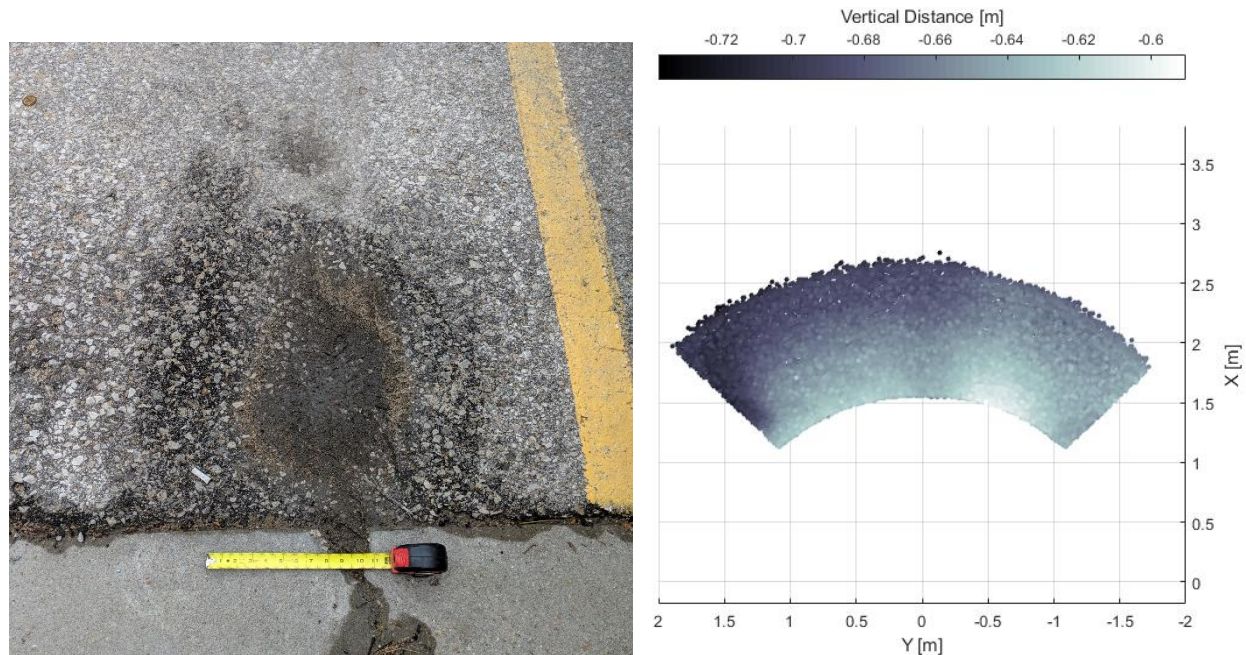
The system was positioned, turned on, and left alone until it completed scanning the pothole and its surrounding road surface. Then, as before, the lidar distance and motor position data, along with the orientation of the system were modeled in MatLab to produce a rendering of the pothole (Figure 3.10). The model of this pothole shows the lower parts in the darker areas. The single straight edge on the left side of the pothole can also be seen on the left side of the model in Figure 3.10. However, the pothole is shallow, and lacks hard edges. Therefore, the rest of the model is difficult to match to the pothole.



**Figure 3.10** Scanned pothole, with one-foot reference (left) and modeled pothole (right).

The second pothole modeled had a smooth bowl shape in the pavement. While this pothole is deeper than the first, it still lacks definite edges. As a result, the model of the second pothole shows the dramatic change in pavement surface closer to the lidar system. However,

further away, the changes in the surface become less apparent as the lidar signal interacts with the pavement at a shallower angle (Figure 3.11).



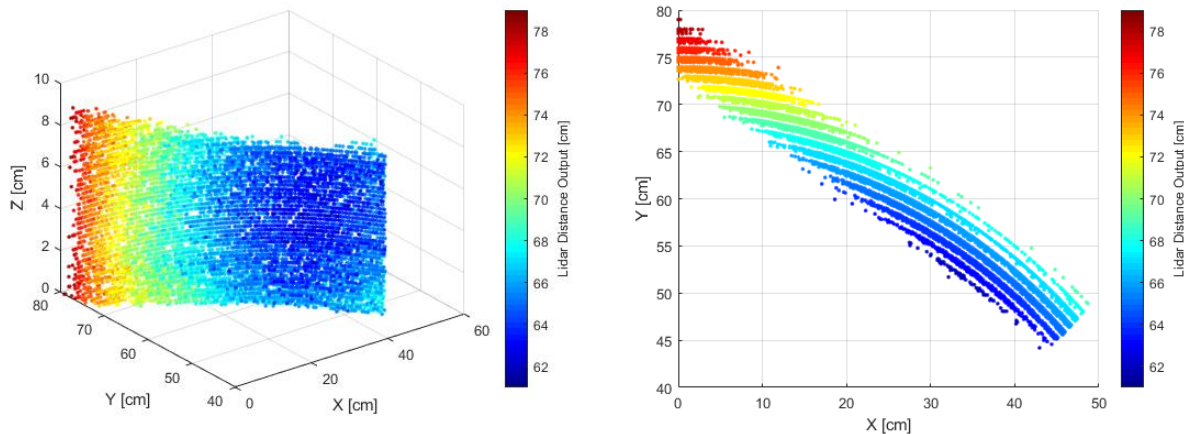
**Figure 3.11** Second scanned pothole with one-foot reference (left) and modeled pothole (right).

Several issues prevented modeling a clear and distinct pothole. The first potholes tested are either shallow (one to two inches deep) or have smooth sides without clear edge definitions. In addition, the scale of the models, as calculated from the lidar rangefinder's output in centimeters, shows 2 m long potholes between 10 and 20 cm deep. While these tests show the lidar system can highlight the comparative differences in surface elevation, a deeper pothole with clear edges would result in cleaner models. Of note, the lidar system was setup between two and four feet away from the pothole at a height of 15 inches so it would not be in a location where a vehicle could hit the system. However, this hindered detailed data as the shallow incident angle to the pavement surface and the inherent inaccuracies of the lidar rangefinder results in scattered data.

Any data point could be closer or farther than the model indicates, hiding some details of the pothole.

### 3.5 Performance Optimization

To create a cleaner model of a pothole, several aspects of the lidar system and testing procedures were adjusted. The first alteration was to the data acquisition setting of the Garmin lidar rangefinder. The optimum setting will provide the most precise set of data at a distance between one and two feet. There are six operational settings pre-programmed into the Garmin Lidar Lite v3 Arduino library. These configurations alter the maximum number of signal acquisitions during measurement, the data acquisition mode, and the detection threshold for a valid measurement [27, 50]. The previous models were created with configuration one as it is suited for short range data collection. To determine the optimum performance settings for the lidar system in a close-range situation, all six settings were used to model the same portion of a wall. While no configuration can provide perfect data, the optimum setting will most closely create a model of a single, smooth, and thin surface with a minimal scattering of data points (Figure 3.12).

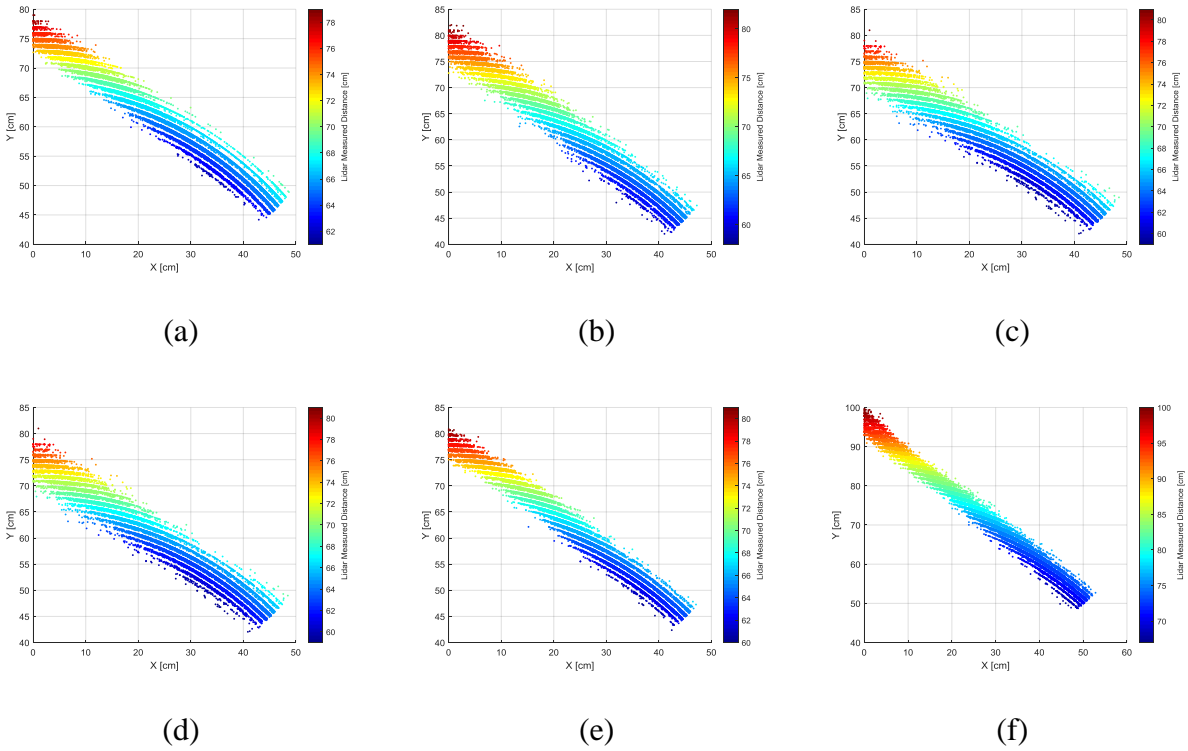


**Figure 3.12** Configuration 0: Default isometric view (left), top view (right) of test wall

The default configuration was designed for a balanced performance and all the following settings have only one difference in programming from the default. Configuration one is for short range, and high-speed performance. Configuration two can switch between the default setting and operating faster at close distances but sacrifices accuracy. Maximum range is the goal for configuration three. Configurations four and five are set for high and low sensitivity, respectively.

Overall, the results show that configuration four had the thinnest wall surface (Figure 3.13). When viewed from above, the models have nearly identical data point spreads. However, the difference between each configuration becomes apparent when comparing the spread of data along the *y*-axis. Configuration four has data ranging from 75 to 80 cm along the *y*-axis, but all the other configurations are spread over about 10 cm. In addition, the data are more compact and less scattered in configuration four when compared to the other settings. The first four tests resulted in a nearly identical scattering of data points and all had set the detection threshold to zero. According to the operational manual for the Garmin Lidar Lite v3, the detection threshold bypass level refers to the amount a signal must peak above the noise floor to be considered a valid measurement [27]. Configurations four and five use the default values for data acquisition count and the collection mode, but are the only configurations that increase the threshold bypass level from zero. This reduced sensitivity results in the rangefinder ignoring weaker return signals while focusing on stronger, clearer, and unambiguous measurements [27, 50]. This more sensitive configuration should be able to better detect the surface differences in and around a pothole.





**Figure 3.13** Top view of test wall model with configuration zero/default (a), one/short range (b), two/switching modes, (c), three/maximum range (d), four/high sensitivity (e), and five/low sensitivity (f)

Given the most precise lidar rangefinder configuration, it is respectively easier to determine any necessary distance correction calculations. The lidar rangefinder is supposed to give the measured distance in centimeters, but the previous pothole models show that the scale was too large. A simple test of the lidar rangefinder showed how the measurements relate to actual distances and if the angle of incidence has a significant influence. The lidar system was set at 14, 18, 22, and 26 inches away from a wall at incident angles of 0, 10, 20, 30, and 40°. At these close distances, the lidar rangefinder is consistently adding almost 10 cm to the actual distance (Table 3.1). Therefore, the lidar data can be readily corrected with a linear adjustment without the need to account for the angle of incidence. The linear regression resulted in a nearly

one-to-one lidar measurement to centimeter scale, but shifted down about 8 cm (Equation 3.1). A similar test using the default configuration during the previous data collections showed identical results.

	Median Lidar Measurement with Configuration Four [cm]				
Distance to Wall [in (cm)]	0° Angle of Incidence	10° Angle of Incidence	20° Angle of Incidence	30° Angle of Incidence	40° Angle of Incidence
14 (35.56)	45.5	44.5	43.5	41.5	43.0
18 (45.72)	55.5	53.5	54.0	53.0	53.0
22 (55.88)	65.0	64.5	65.0	66.0	62.5
26 (66.04)	75.5	73.5	73.0	73.5	72.5

**Table 3.1** Lidar system measurements at various distances and incident angles

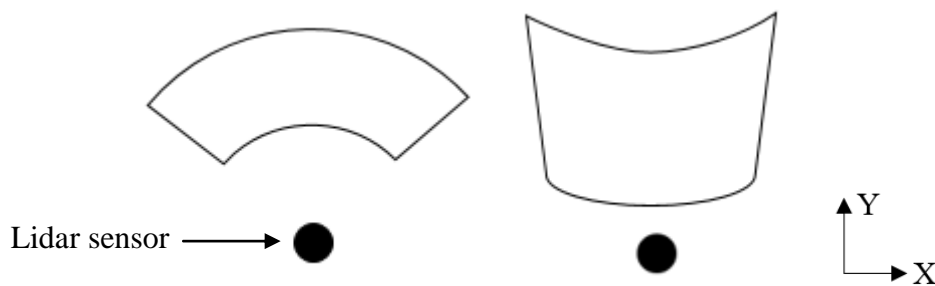
$$d = 0.996(L) - 7.905$$

**Equation 3.1** Lidar measurement,  $L$ , to accurate distance,  $d$ , in centimeters

In addition to changing the lidar rangefinder configuration in the Arduino code, the system's orientation and position is now recorded in the electrically erasable programmable read-only memory (EEPROM) of the Arduino microcontroller. The EEPROM has the ability to store data without a power supply [51]. This change is a precaution to prevent data misrepresentation in case of any unpredicted power loss. Previously, any power loss triggered a program restart and created a secondary origin point and that could alter the intended field of view. By saving the position of both motors, as well as the direction of rotation to the EEPROM at each data point, no secondary origin point is created during an unexpected program restart. However, as before,

saving data takes time and slows the overall mapping speed of the lidar system. Due to this, the system requires one hour to collect almost 131,000 data points.

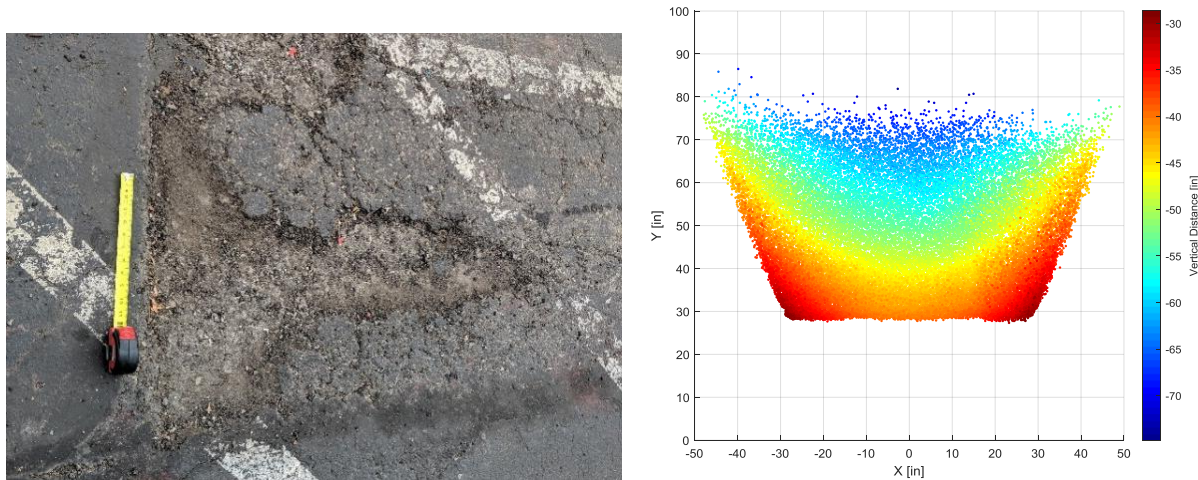
Initially, the model relied on a MatLab function for changing spherical data to Cartesian coordinates. In a top-down view, the spread of data points is shaped in an arc around the lidar sensor. While this is correct for tests when the lidar system is level, during the first pothole test, the lidar system was positioned on its side to achieve a better range of motion. When in a level position, the lidar rangefinder first rotates along the global  $z$ -axis and then rotates along a changing line in the global  $x$ - $y$  plane. However, when set on its side, the rangefinder will rotate along the global  $y$ -axis and then a changing axis in the global  $x$ - $z$  plane. As a result, the corrected data scan is more rectangular and curves away from the lidar system (Figure 3.14).



**Figure 3.14** Resulting scan outlines with level system (left) and 90° system orientation (right)

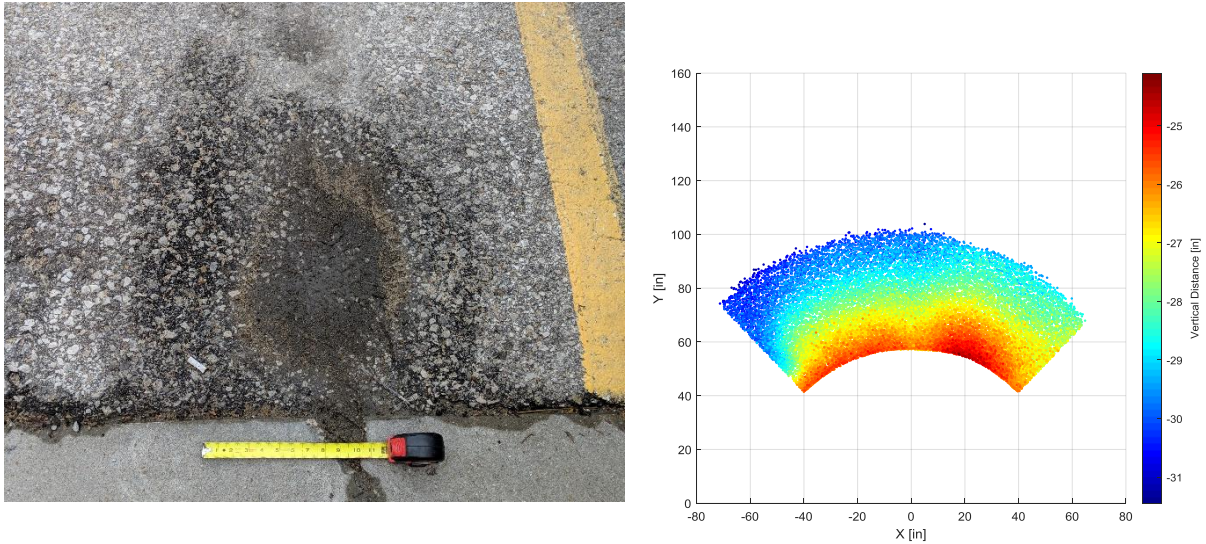
When the distance and orientation calculations were corrected for the previous models, the results showed little definition. Here, the first pothole model now finds a smooth surface (Figure 3.15). The possible details of the pothole seen in the previous modeling attempt are no longer apparent with the improved orientation and distance calculations. While discouraging, this result is not unexpected. The first pothole has a high length to depth ratio and the lidar system was positioned over two feet away. Neither condition was conducive to creating a detailed model. As shown, the data collected becomes more scattered as the lidar system is angled closer

to parallel with the pavement surface. The near edge of the scan, shown in red and orange, is a clean line of data points because the lidar rangefinder is at a steep angle of incidence. However, the far edge, shown in blue, has more scattered data because the shallower angle is more sensitive to both surface variations and lidar accuracy decreases with distance [27]. This results in the lidar system being unable to distinguish between the slight changes in surface elevation.



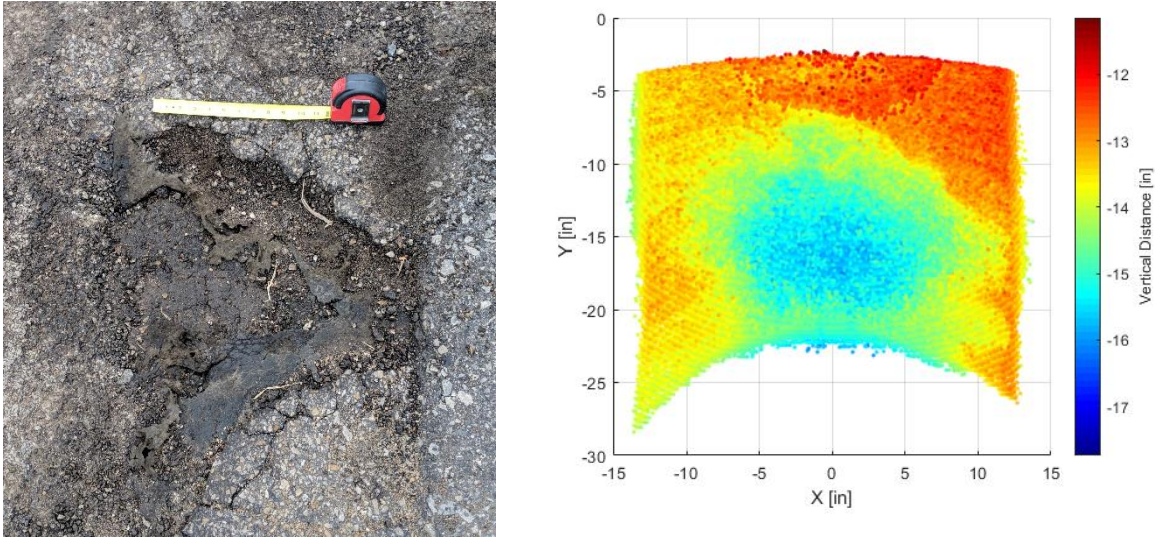
**Figure 3.15** First pothole with corrected distance and orientation calculations

The second pothole modeled only required alterations to the distance calculation since the lidar system was in a level position during data collection. The new model shows the same changes in elevation, but at an accurate scale (Figure 3.16). Here, both tests continue to suffer from a lack of detail due to the distance between the potholes and the lidar system, as well as undefined edges with obvious differences in surface elevation.



**Figure 3.16** Second pothole with corrected distance calculation

Based on a better understanding of the lidar system, orientation, and distance corrections, a new pothole was found for data collection. When compared to the previous tests, this pothole has larger, deeper, more defined edges, and is surrounded by flat and level pavement (Figure 3.17). In addition, the lidar system is positioned a few inches away from the edge of the pothole to limit lidar measurement errors and keep data within the previously determined distance calculation (Equation 3.1). Furthermore, the calculation of the Cartesian  $x$ ,  $y$ , and  $z$  data points from the spherical information collected was corrected from the initial tests. With the system and testing conditions optimized, the resulting model shows a clear pothole definition, with an accurate shape and scale (Figure 3.17).



**Figure 3.17** New pothole with one-foot scale (left) and improved model (right)

One flaw with the lidar system and model after optimization remains the processing speed of the Arduino microcontroller. The microcontroller has a 16 MHz clock speed and the additional use of the EEPROM for data storage further slows the overall lidar system operating speed [32].

### 3.6 System Diagnosis

While the 3-D lidar system can identify more dramatic changes in the pavement surface, data collection is time-consuming. Each pothole requires the system to remain motionless for nearly one hour. To boost data collection speed, one could increase the step angle of the motors between each data point. This will decrease the sweep time, but also reduce the data density. Moreover, if the battery packs in the lidar system are fully charged, there is minimal risk of the system shutting off momentarily from low power; hence, the EEPROM data storage is not needed and can be removed from the Arduino microcontroller code. This would increase the system's operating speed from roughly 35 data points per second to nearly 70 data points per

second. Another option to increase data collection speed is to implement a faster data processor in the microcontroller. The Arduino Mega 2560 microcontroller has an operating speed of 16 MHz [32]. Instead, a faster microprocessor, such as the Raspberry Pi 4B, which has a 1.5 GHz processor, would be able to map potholes in a fraction of the time [52].

Finally, if the 3-D lidar system is needed to map potholes while moving, an accelerometer should be added to the system. With the current lidar system configuration, there is no method of modeling system motion. Therefore, if the system is moved during data collection, the entire data set becomes skewed and is likely unusable. Hence, an accelerometer will aid the model in determining where each data point is located in relation to the other data points.

### 3.7 Commercial Systems

The viability of the lidar systems constructed also depends on how they compare to commercially available lidar systems. These systems are not designed specifically for use with electric bicycles or to model potholes in the road. Instead, these commercial systems are used in a variety of industries: mining, driver assist, autonomous vehicles, property mapping and planning, construction, and wildlife management [23, 25, 53-58]. Many of these available systems have exponentially more powerful lasers and faster processing speeds than the systems described in this thesis. As such, the commercial systems, in general, have greater distancing capabilities and can collect upwards of tens of thousands of data points per second. However, this high performance also comes with a significant price. Not one of the commercially available systems had a listed price tag and when contacted directly, only one manufacture was able to give a quote for nearly \$43,000, which presumably is the price range of the other commercial

systems. The constructed systems, while not nearly as powerful, only costs around \$300 each for the components (Table 3.2).

System	Type	Range [ft/m]	Accuracy	Data points per second	Weight [kg]	Unique Features
3 <sup>rd</sup> Gen. 2-D Lidar System [31]	2-D	197 / 60	+/- 1.5 in or +/- 1.5%	25	0.59	~\$300
Phoenix Lidar systems Scout-16 [57]	2-D	131 / 40	+/- 5.5 cm	300,000	1.65	16 lasers
YellowScan Surveyor [58]	2-D	164 / 50	+/- 5 cm	300,000	1.6	Uses Velodyne lidar
Slamtec RPLIDAR A1 [56]	2-D	39.4 / 12	+/- 0.2 cm	8,000	0.17	
LeddarTech Pixell [53]	2-D	134.5 / 41	+/- 5 cm	20	2.25	Flash Illumination
2 <sup>nd</sup> Gen. 3-D Lidar System [27]	3-D	131 / 40	+/- 1 in	35 (to 70)	0.66	~\$300, 360°×90° field of view
RedTail Lidar RTL-400 [55]	3-D	393.7 / 120	+/- 1.5 cm	1,000,000	2.13	40°×40° field of view



System	Type	Range [ft/m]	Accuracy	Data points per second	Weight [kg]	Unique Features
Paracosm PX-80 [54]	3-D	328 / 100	+/- 3 cm	300,000	2.9	360°×30° field of view, uses Velodyne lidar

**Table 3.2** Comparison of created lidar systems and commercially available lidar systems

### 4.1 Introduction

As previously mentioned, bicyclists are more likely to be injured or killed in accidents than vehicle passengers when comparing both distance and time traveled [9, 10]. For instance, a cyclist is ten times more likely to die and has a fifty times greater possibility of injury than a vehicle passenger [10]. Statistically, a majority of bicycle fatalities occur when a moving vehicle strikes a traveling bicycle from behind during rush hours [11]. Unfortunately, many safety features common in vehicles that successfully protect passengers (e.g., seatbelts, airbags, and roll cages) cannot be applied to bicycles. Therefore, the best protection for bicycle riders is to reduce or prevent crashes before they can happen [12].

While a lidar system, as shown in a previous chapter, can be used to alert riders of incoming vehicles and other unseen dangers, if the rider is physically unable to move out of the way fast enough, they will still crash. Here, a typical bicycling commuter will never be able to travel at the same speed as vehicles on the road and will have a slower and unsteady acceleration. However, an e-bike can reach higher speeds more readily and will be able to merge and move with traffic without a mechanical input from the rider. Additionally, the rider of the e-bike will not tire or be distracted and can focus more completely on their surroundings.

While an e-bike has the potential to reduce accidents, if the battery pack runs out of power during the commute the risk of injury immediately increases again. Therefore, it is important to accurately determine if a commute is feasible on a single charge. It would also be possible to monitor the motor torque and speed to allow for the e-bike to operate in the optimal motor efficiency range whenever possible. Additionally, the net force of the e-bike can alert the rider to necessary stopping distance and risk of injury in the event of a collision. Furthermore,

tracking the speed, acceleration, and direction of the e-bike, could improve blind-spot vehicle detection in an integrated lidar system and reduce false positives as mentioned in Chapter 2.

Modeling the route and power consumption of an e-bike is a similar process to simulating other electric vehicles. Road inclination, wind resistance and incoming angle, total weight, rider size, tire traction, and acceleration are all vital external factors to model power consumption and the forces involved. Additionally, the design of the e-bike (i.e., electric motor capabilities, battery pack size, and auxiliary systems) also influences the weight of the e-bike and the instantaneous velocity and acceleration characteristics; hence, they must be included in the e-bike model.

## 4.2 Forces Modeling

When investigating the motion of an e-bike, it is important to start with the external forces acting upon and caused by the e-bike. In specific, the e-bike exerts a tractive force,  $F_T$ , to overcome the forces resisting the direction of motion ( $F_D$ ,  $F_G$ , and  $F_R$ ) resulting in a net acceleration force,  $F_x$ . This is based on Newton's Second Law of Motion as follows [59]:

$$F_x = F_T - F_D - F_G - F_R$$

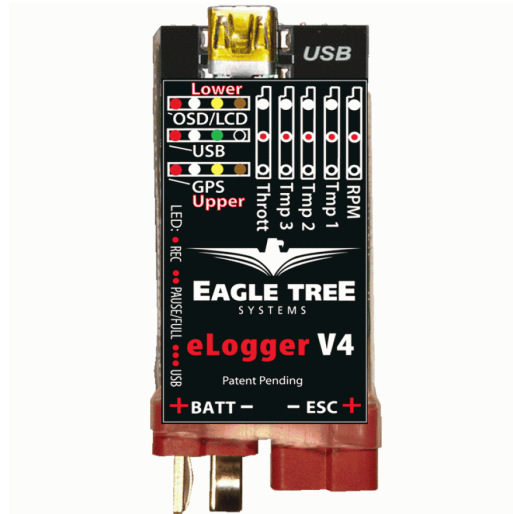
**Equation 4.1** Newton's Second Law summation of forces acting on the e-bike.

This net acceleration force of the e-bike is equal to the mass,  $m$ , multiplied by its acceleration (Equation 4.2). During testing, both the mass and the acceleration can be monitored. The mass of the bike and rider is measured before testing, and the speed,  $v$ , of the e-bike is recorded using an Eagle Tree Systems E-logger v4 as shown in Figure 4.1 [60]. Therefore, the

acceleration force is known through the entire test as the logger will record velocity of the e-bike and time.

$$F_x = m \frac{dv}{dt}$$

**Equation 4.2** Net acceleration force determined with bike and rider mass and measured acceleration.



**Figure 4.1** Eagle Tree eLogger v4 used on the e-bike to record data for the model.

The drag force is the next calculable force in Equation 4.1 [59]. The basic drag equation focuses on air resistance caused by an object as it moves at a known speed (Equation 4.3). The air density,  $\rho$ , is determined using the ideal gas law, the air temperature, and pressure during testing. Frontal area,  $A_f$ , changes depending on the size of the rider, sitting position, and direction of wind [61-63].  $C_D$ , the coefficient of drag is difficult to estimate without testing as it changes with bike and rider size, shape, clothing, and sitting position [62, 64-66]. Given the complexities of calculating  $A_f$  and  $C_D$  individually, many researchers combine them into a single constant effective frontal term [61, 64].

$$F_D = \frac{1}{2} \rho A_f C_D v^2$$

**Equation 4.3** Generic drag force acting on a moving body caused by air resistance.

While this drag equation works well in controlled settings like wind tunnels, a more realistic model must account for drag caused by wind as well as air resistance. For a bicycle, the wind has a dramatic effect on the drag experienced by the rider. Equation 4.4 includes the component of wind acting tangent to the direction of motion of the body [59]. The wind speed,  $v_{wind}$ , is positive when acting as a tailwind and negative when impacting the rider via a headwind. The acute angle between the direction of the wind and the direction of the vehicle motion is  $\phi$ .

$$F_D = \frac{1}{2} \rho A_f C_D (v - v_{wind} \cos \phi)^2$$

**Equation 4.4** Force of drag acting on vehicle caused by air and wind resistance.

However, even a side wind acting exactly perpendicular to the direction of motion will slow a bicycle. Equation 4.4 shows that a perpendicular wind has no component acting with or against the direction of motion. This is most likely due to the difference in the body geometry of a vehicle and that of a person on a bicycle. A perpendicular crosswind acting on a vehicle hits a relatively flat side and has little effect against the direction of motion. However, a person has a respectively rounder shape and a crosswind will cause small amounts of drag [67, 68].

A second version of the drag force equation to include wind resistance was developed with the shape of a bike and rider in mind (Equation 4.5) [68]. Adding the wind and bike direction vectors results in a singular apparent wind speed,  $v_{app}$ , from one direction at an acute angle  $\beta$  to the direction of the bike. Additionally, this equation has a directivity function,  $\lambda$ , to approximate the effective drag area as the apparent wind angle changes (Equation 4.6). This

function results in an adjustment factor for the effective drag area ranging from one (frontal) to typically around 1.2 (side) [68].

$$F_D = \frac{1}{2} \rho [\lambda (A_f C_D)] v_{app}^2 \cos \beta$$

**Equation 4.5** Drag force acting on e-bike with wind resistance and adjusted effective drag area.

$$\lambda = \cos^2 \beta + \frac{A_{side} C_{D,side}}{A_{front} C_{D,front}} \sin^2 \beta$$

**Equation 4.6** Directivity function to adjust effective drag area based on apparent wind angle [68].

Since the drag force represents between 56 to 96% of the total resistance acting on a bicycle, it is important to model this force as accurately as possible [68, 69]. A final factor regarding the drag force on a bike includes the air resistance on the spokes of the wheels as they rotate. However, this spoke drag was determined to be on a scale of one thousandth of the drag force acting on the body of the bike and rider [69]. Therefore, it will not be considered here as it should not noticeably affect or improve model accuracy. Additionally, the effective drag area,  $A_f C_D$ , will be assumed constant. As the frontal area, air density, and bike speed measured can be measured, the coefficient of drag for a commuter cyclist sitting upright is estimated to be 1.1 [62, 64-66].

The next most influential resistive force acting on the e-bike is the gradation force,  $F_G$ , working with or against gravity,  $g$ , when riding uphill or downhill (Equation 4.7) [59]. This force will be negative when moving downhill as the rider and electric motor do not need to supply as much power to maintain a constant speed. The slope of the route,  $\theta$ , can be calculated from

measured data during testing. Therefore, the gradation force is subject to far less uncertainty and estimation than the drag force.

$$F_G = mg \sin \theta$$

**Equation 4.7** Gradation force accounting for slope of route.

The final resistive force acting on the e-bike is the rolling resistance force (Equation 4.8). Several factors affect the coefficient of rolling resistance,  $\mu_r$ , such as ground surface texture, tire width, tire pressure, and tire tread pattern. In general, the more surface area in contact between the tires and the ground surface, the larger the coefficient of rolling resistance. Narrow, street racing tires have a coefficient of roughly 0.0033, standard tires are slightly higher around 0.007, and off-road, studded tires have the highest coefficient ranging from 0.013 to 0.017 [62, 69].

$$F_R = \mu_r mg$$

**Equation 4.8** Rolling resistance force with a constant, estimated coefficient of rolling resistance.

The only mechanical input from the rider and/or the electric motor causes a tractive force to move the e-bike forward. As shown in Equation 4.1, the tractive force,  $F_T$ , equals the sum of all resistive forces and the net acceleration force. However, it is difficult to identify how much power is generated by the rider and the motor if both are working simultaneously. Therefore, to simplify data collection and modeling, only the electric motor will be used to move the e-bike during testing.

### 4.3 Motor Modeling

Once the resistive forces acting on the e-bike and the motor force are determined, it is a relatively straight-forward process to determine the power the motor requires from the battery pack. As the tractive force is already known, the torque exerted by the rear wheel,  $\tau_w$ , can be found readily by measuring the radius of the wheel,  $r_t$  (Equation 4.9) [59]. Standard bicycle wheels and the e-bike wheels were measured to have a radius of 12.25 inches (0.311 m).

$$\tau_w = F_T r_t$$

**Equation 4.9** Wheel torque corresponding to the tractive force.

For cars and other vehicles, a second equation is typically required to determine the torque exerted by the motor or engine. This is because transmission gears,  $i_0$  and  $i_g$ , and a driveline efficiency,  $\eta_t$ , usually exist to transfer power from the motor or engine to the wheels of the vehicle. However, on this e-bike, the motor is attached directly onto the rear wheel (i.e., direct drive motor). Therefore, no gears exist between the motor and the wheel and theoretically, no power is lost through a non-existent driveline. As a result, the wheel torque will be equivalent to the brake torque,  $\tau_b$ , caused by the electric motor [59]:

$$\tau_b = \frac{\tau_w}{i_0 i_g \eta_t} = \tau_w$$

**Equation 4.10** Wheel torque to brake torque with motor directly on rear wheel.

The rotational speed,  $N$ , of the electric motor can be calculated from the speed data of the e-bike previously mentioned. Again, vehicles usually must take the transmission into account to determine a corresponding rotational speed of the engine or motor; however, this is irrelevant in



this instance. Therefore, the linear speed of the e-bike can be directly related to the rotational speed of the motor in revolutions per minute [59]:

$$N = \frac{v}{2\pi r_t}$$

**Equation 4.11** Motor speed calculated based on linear e-bike speed and tire radius.

With the brake torque and the motor speed calculated, it is possible to determine the brake power,  $P_b$ , the motor exerts to the e-bike [59]:

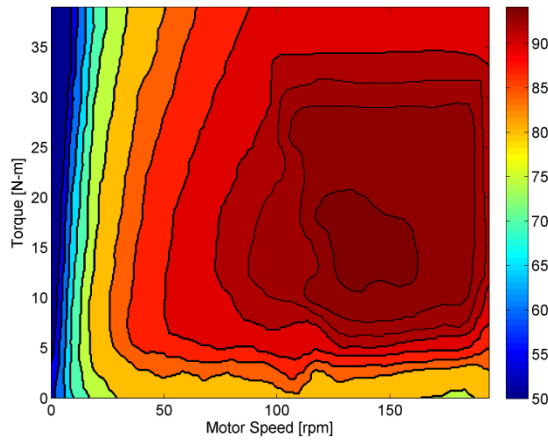
$$P_b = 2\pi\tau_b N$$

**Equation 4.12** Brake power output from the electric motor.

Finally, to determine the power the motor requires,  $P_m$ , from the battery pack (Equation 4.13), one must first find the motor efficiency [59]. However, the motor efficiency,  $\eta_m$ , changes with both the brake torque and motor speed. In general, motor efficiency maps exist for different types and sizes of motors; unfortunately, the exact motor map used on the e-bike is not readily available. Therefore, a generic efficiency map (Figure 4.2) for a different brushless direct current (BLDC) motor is scaled down to match the maximum rotating speed and torque output capabilities of the Heavy-Duty BLDC motor employed in the e-bike [70].

$$P_m = \frac{P_b}{\eta_m}$$

**Equation 4.13** Motor input power using the power output and motor efficiency.



**Figure 4.2** BLDC motor efficiency map, scaled to the size of motor used in the e-bike [71].

#### 4.4 Battery Modeling

After calculating the power draw to the motor and recording the voltage, current, and temperature of the battery pack, it is possible to model its performance throughout the testing process. It is important to note that the total power draw from the battery pack does not exclusively go to the electric motor. The e-bike has a data recording system that monitors the battery pack current and voltage, with a connected GPS for speed, elevation, and distance. As the recording system has no need to vary power consumption during testing, it is assumed to be a constant draw of 0.1 Amp [60, 72, 73]. Additionally, the Eagle Tree data recorder installed on the e-bike will record the current draw and voltage of the battery pack. Therefore, Equation 4.14 will serve as a verification to compare the measured current,  $I$ , and voltage,  $V$ , of the battery pack against the calculated values for motor and auxiliary power,  $P_{aux}$ .

$$P_m + P_{aux} = I \cdot V$$

**Equation 4.14** Total calculated power draw.

Once the power consumption is verified, the next step is to model the battery pack behavior. In addition to recording its current and voltage, the data logger can record the temperature,  $T$ , of the battery pack throughout testing. This is important as it is known that temperature affects a battery cell's ability to supply power and can be damaged at extreme temperatures. Furthermore, with a known battery chemistry, it is possible to use the Hausmann-Depcik model for battery capacity,  $Q$  (Equation 4.15). The three constants,  $\gamma$ ,  $\chi$ , and  $\delta$ , are set for several common battery chemistries including the lithium nickel manganese cobalt oxide (NMC) cells used with the e-bike. Additionally, the reference values,  $I_{ref}$  and  $T_{ref}$ , are set to 1 A and 298 K, respectively [74].

$$Q_{t+1} = Q_t - \gamma \left( \frac{I_t}{I_{ref}} \right)^\chi \left( \frac{T_{ref}}{T_t} \right)^\delta \Delta t$$

**Equation 4.15** Peukert capacity model expanded to include temperature effects [74].

Finally, the *SOC* of the battery pack, which explains how much charge is left in the pack, can be determined throughout testing. *SOC* is calculated by comparing the present capacity to the nominal capacity,  $Q_{nom}$  (Equation 4.16). Rechargeable batteries typically perform best when they are not fully charged or discharged but when kept between roughly 20 and 80% *SOC*.

$$SOC_t = \frac{Q_t}{Q_{nom}}$$

**Equation 4.16** State of charge calculation using battery pack capacity.

#### 4.5 Model Results

To test the model prior to collecting data with the e-bike, the simplest scenario is represented: assuming a constant speed on a level ground, without headwind or crosswind, and

the temperature of the battery pack remains constant (Table 4.1). Therefore, the forces acting on the e-bike and the battery pack power consumption remains constant. Here, several variables in this test of the model are estimates. The bike rider and mass are estimated using data from the design and construction of the e-bike [29]. The mass along with air density, frontal area, drag coefficient, and battery temperature are estimated and their values will be measured prior to data collection.

Parameter	Value
Speed	8.9 m/s (20 mph)
Bike and Rider Mass	88.23 kg (194.5 lbs)
Air Density	1.24 kg/m <sup>3</sup>
Frontal Area	0.3 m <sup>2</sup>
Wind Speed	0 m/s
Drag Coefficient	1.1
Rolling Resistance Coefficient	0.007
Tire radius	0.311 m
Road Angle	0°
Battery Temperature	300 K (80 °F)
Battery Nominal Capacity	10 A·h / 360 W·hr

**Table 4.1** Values for respectively simple modeling scenario.

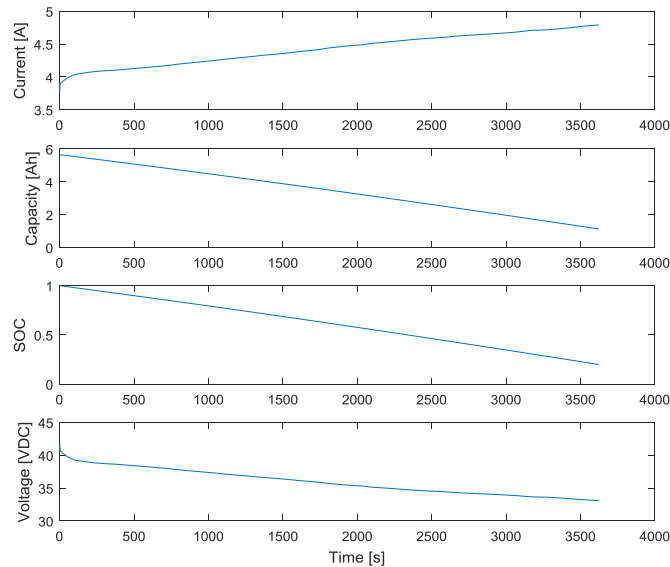
Using data readily available for lithium nickel cobalt aluminum oxide, the model starts at 100% and ends when the battery pack reaches 20% SOC [59]. It is important to note the batteries on the e-bike are believed to be NMC and the final model will use the corresponding data.

However, for the purpose of testing the model, the exact chemistry of the lithium battery pack is not needed. Running this scenario results in the e-bike having a range of 20.05 miles and lasting 60.4 minutes on a single charge (Table 4.2). Additionally, by comparing the nominal watt-hours of the battery pack to the calculated power used, the runtime is estimated to be 78.4 minutes ending at 20% SOC. The battery pack current, capacity, SOC, and voltage change during the modeled situation are shown in Figure 4.3.

Model Variable	Value
Drag force [N]	16.206
Gradation force [N]	0
Rolling resistance force [N]	6.059
Acceleration force [N]	0
Tractive force [N]	22.265
Braking torque [Nm]	6.925
Motor speed [rpm]	273.3
Motor efficiency [%]	90
Motor power [W]	220.2

**Table 4.2** Calculated model variable remaining constant throughout scenario (

## Appendix B).



**Figure 4.3** Resulting battery pack data from modeled scenario.

In Figure 4.3, as the pack loses charge, its voltage drops as to be expected. Since the power draw is constant, the amperage increases to compensate for this loss of voltage. The battery data in this figure far smoother than it would be during real data collection as the power draw to the motor will vary dramatically to account for e-bike acceleration, changing wind drag, going uphill, and coasting downhill. Crucially, real-world data collection and modeling in the following chapter will show how the varying forces acting on the e-bike will have a noticeable effect on the motor performance, battery pack charge, and overall e-bike driving range.

### 5.1 Introduction

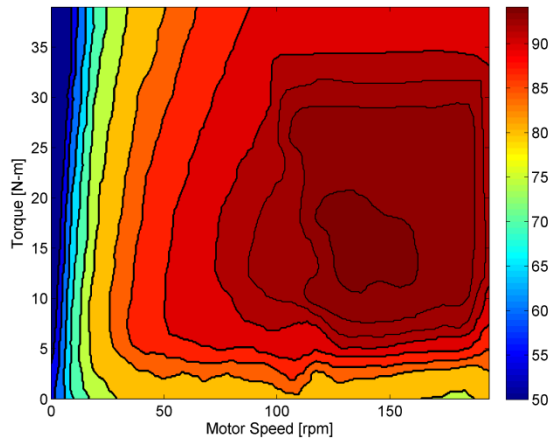
With the theoretical equations established and the variables needed to model the e-bike identified in Chapter 4, data collection can commence. To estimate the voltage, current, and SOC of the battery pack, the model requires a record of the e-bike speed, altitude, wind speed, wind direction, battery pack temperature, and time throughout testing. Additionally, the initial pack voltage, total weight of rider and e-bike, frontal area of rider and e-bike, tire radius, and air density must be measured just prior to the start of each test. The e-bike used for testing was built by former students with this type of data collection in mind and is heavier than a commercially available e-bike. A small shelf over the rear wheel hold the battery pack, data recorder, and GPS sensor while a separate wind sensor is mounted onto the front handlebars.

### 5.2 Data Collection

For modeling, several properties are measured before testing. A female rider, 5'7" tall and 120 pounds sits on a 65.2 pound e-bike that has a frontal area at 0.19 square meters, measured by comparing the size of the rider and e-bike to a known reference area in a photograph using imaging processing in MatLab. As previously mentioned, the tire radius is 0.311 meters while the drag coefficient and rolling resistance coefficient are estimated to be 1.1 and 0.007, respectively (Table 5.1). The motor efficiency is also calculated using the BLDC motor map scaled to the manufacture's specifications (Figure 5.1). Additionally, air temperature, air pressure, and wind direction are recorded to calculate the drag force noted before each test (Table 5.2).

Parameter	Measured or Estimated Value
Rider Mass [kg]	54.53
Bike Mass [kg]	29.57
Frontal Area [m <sup>2</sup> ]	0.19
Coefficient of Drag [-]	1.1
Coefficient of Rolling Resistance [-]	0.007
Tire Radius [m]	0.311
Nominal Battery Capacity [Ah]	10
Maximum Battery Voltage [VDC]	42

**Table 5.1** Physical properties of combined e-bike and female rider



**Figure 5.1** BLDC motor efficiency map, scaled to the size of motor used in the e-bike [71].



Parameter	Route 1	Route 2
Wind Speed [m/s]	3.13	3.58
Wind Direction (North is 0°)	135	135
Air Temperature [°F]	77	81
Air Pressure [inHg]	30	30

**Table 5.2** Weather conditions recorded for e-bike test rides.

The electric bicycle has several devices for recording data during testing (Table 5.3). An Eagle Tree Micro GPS Expander v4 measures the latitude, longitude, elevation, speed, and direction of motion [73]. It connects to the Eagle Tree eLogger v4 that records the data from the GPS and also measures the voltage and temperature of the battery pack, the current running from the batteries to the motor, and the time [72]. Separately, a Modern Device Wind Sensor Rev. P attached to the handlebars of the e-bike records the time and wind speed but is not synchronized with the eLogger [75]. The wind and temperature data from the wind sensor was calibrated using published information compensating for the air temperature and the measured zero wind voltage,  $V_0$ , was 1.346 VDC (Equation 5.1) [76]. To coordinate the collected data, both the wind sensor and the eLogger are set to record at a rate of 10 samples per second. Since the e-bike can start and stop quickly, the beginning and end of the tests are identifiable in the eLogger through sudden changes in current draw and bike speed, and through the wind sensor by notable changes in wind speed. From this, the data can be manually synchronized using the start and end of each test ride.

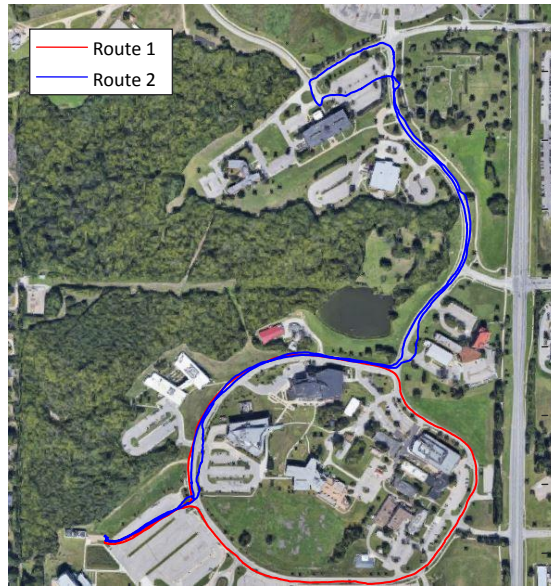
$$v_{wind} = \left( 2.3953 * \frac{0.0049Wind - V_0}{(0.0049Temp - 0.4)^{0.115157}} \right)^{3.009364}$$

**Equation 5.1** Calibration for collected *Wind* and *Temp* data by the wind sensor.

Measurement	Accuracy
GPS coordinates <sup>1</sup>	± 10 <sup>-12</sup>
Altitude <sup>2</sup> [m]	± 0.1
Voltage <sup>3</sup> [VDC]	± 0.01
Current <sup>3</sup> [A]	± 0.01
Time <sup>3</sup> [s]	± 0.01
Wind speed <sup>4</sup> [mph]	± 0.01
Battery temperature <sup>5</sup> [°C]	± 0.1

**Table 5.3** Accuracy of measurements from (1) Eagle Tree Micro GPS Expander v4, (2) GPS visualizer [77], (3) Eagle Tree ELogger v4, (4) Wind Sensor Rev. P, and (5) Eagle Tree Temperature Sensor.

Two different routes were taken during testing (Figure 5.2), both on the same day and involve changes in elevation and speed. During these tests, the wind was approximately 8 mph from the southeast and the temperature was 77 °F. Overall, the e-bike reached a top speed of about 20 mph.



**Figure 5.2** Two routes for data collection across West Campus at the University of Kansas.

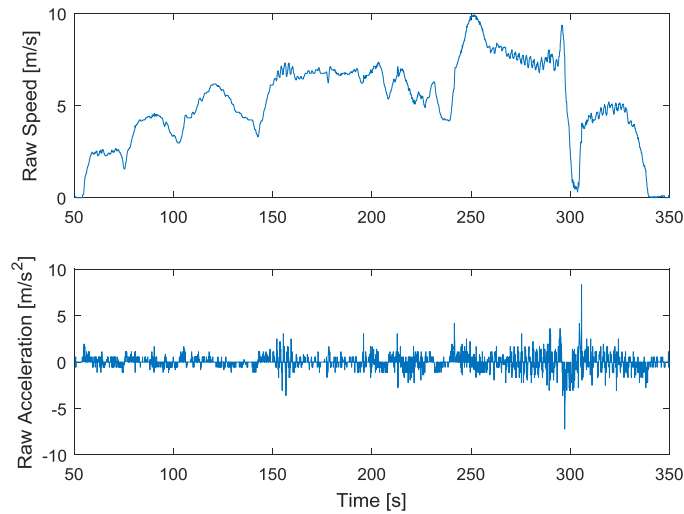
### 5.3 Data Processing

After data collection, a portion of the elevation data was found to be inaccurate. In specific, the GPS requires time to communicate with satellites and capture elevation data [73]. Hence, the altitude data captured initially did not calibrate accurately with the battery pack (i.e., voltage and current) information during the first half of testing. In addition, it is relatively simpler to find accurate elevation information from latitude and longitude data [77]. As a result, a combination approach was taken and the latitude and longitude were verified against the second half of the elevation data provided by the GPS.

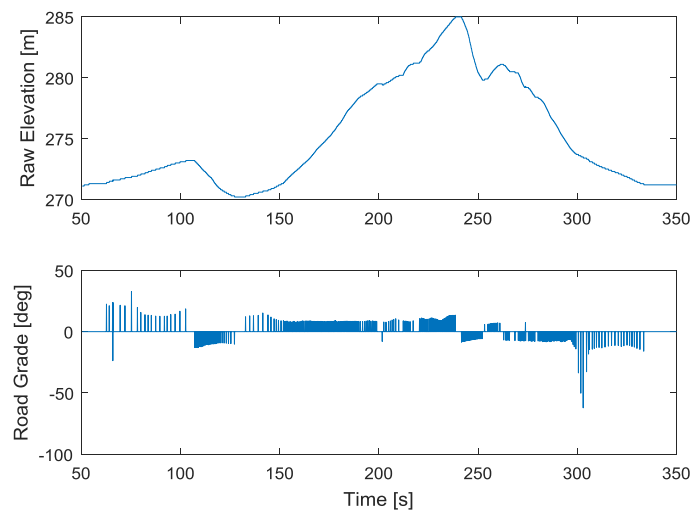
Moreover, it was determined that the raw data needs to be filtered before it can be used with the model. Since digital recording systems were used, there is an unavoidable discretization of data. Specifically, data collected from the GPS only records to the nearest tenth of a meter and meter per second for the elevation and e-bike speed, respectively. Therefore, when calculating the acceleration of the e-bike and road incline, the model illustrates significant oscillations and

inaccurately represents the test ride. For instance, the raw data finds the e-bike accelerating and decelerating rapidly; whereas, the e-bike ride has a relatively smoother motion (Figure 5.3).

Moreover, raw elevation data changes by 0.1 meters from one data point to the next, resulting in sudden and steep road angles before immediately returning to a level road (Figure 5.4).



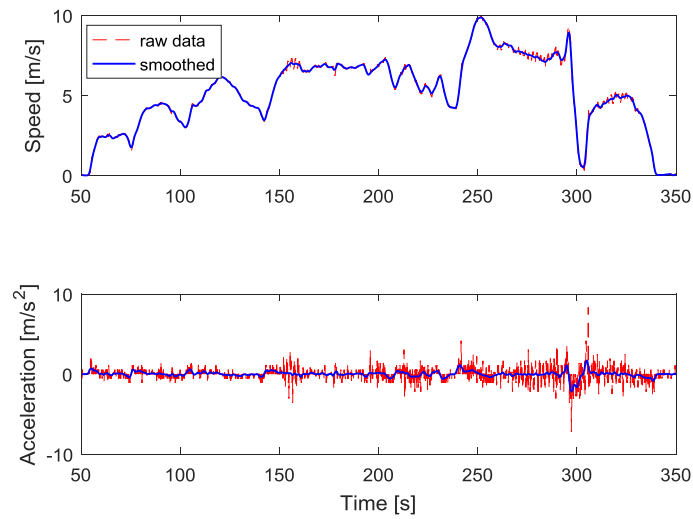
**Figure 5.3** Collected raw e-bike speed data and resulting calculated acceleration.



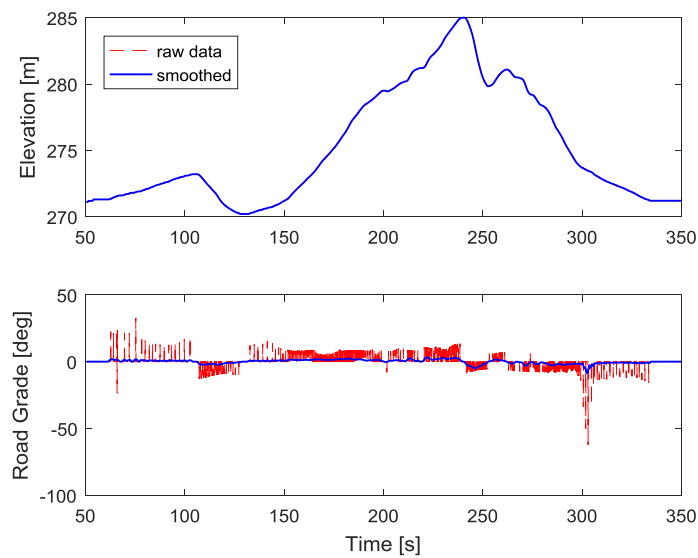
**Figure 5.4** Raw elevation data and resulting road inclination calculation.

Therefore, to model the e-bike accurately, raw data must be smoothed to estimate intermediate values from discrete data. Different methods of smoothing were compared, as well as how broadly they were applied to the raw data. The first method involved a moving average that finds the mean value of all data points in a span to become the new value of the center point in that span. When applied to a small span of data, the resulting data are still visibly discrete but increasing the span removes and flattens notable features of the data. The second method is a local, weighted regression and operates similarly to the moving average method while placing more importance on data closest to the point of interest. Linearly weighted coefficients allow the smoothed data to retain key features of raw data while reducing the effects of oscillating and discretely changing data. Another locally weighted regression tactic uses squared weighted coefficients that more closely follow raw data while retaining discrete changes in data. Robust versions of the weighted coefficient methods exist that eliminate any influence of outlying data points [78]. However, over small spans of data, they are nearly identical to the original regression methods and over larger data spans, removing outliers will remove notable spikes in the raw data.

Overall, using linearly weighted coefficients was the preferred method for smoothing the data while not eliminating dramatic changes. Further testing to optimize the span of the smoothing method shows suitable smoothing over 31 data points at a time. In terms of the e-bike testing, this span analyzes 1.5 seconds before and after the estimated data point. The resulting smoothed data, shown in Figure 5.5 and Figure 5.6, better represents the test rides.



**Figure 5.5** Smoothed GPS speed and resulting acceleration compared to original data

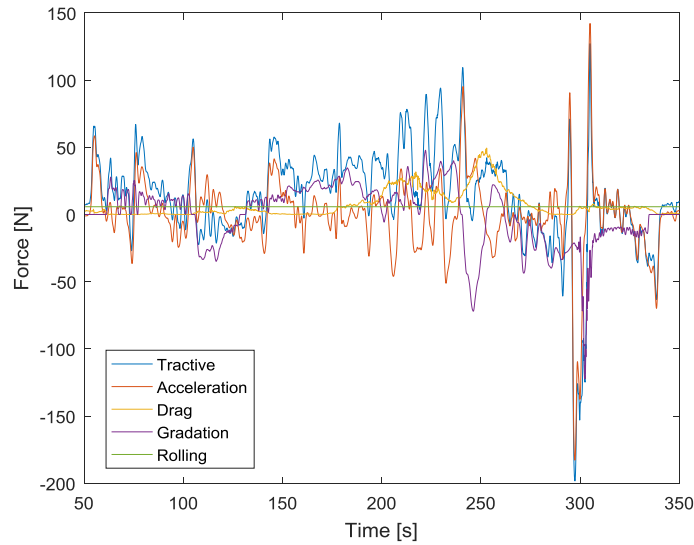


**Figure 5.6** Smoothed GPS elevation and resulting road grade compared to original data

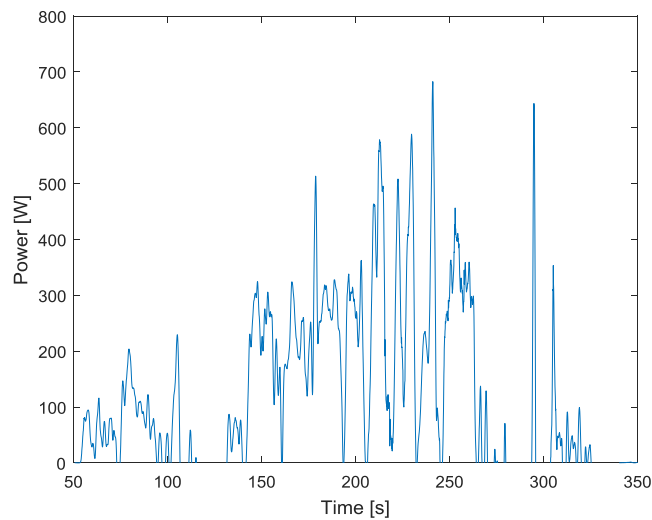
#### 5.4 Data Modeling

Once the data collected from the e-bike was processed, it was applied to the model created in Chapter 4. Using smoothed data results in the force plot shown in Figure 5.7 and the

power drawn by the motor in Figure 5.8. As expected, the gradation and acceleration forces have strong influences on the tractive force that increases as the route goes uphill, followed by a drop when going downhill. It should be noted that the motor power draw excludes negative values as the e-bike is not capable of regenerative braking.

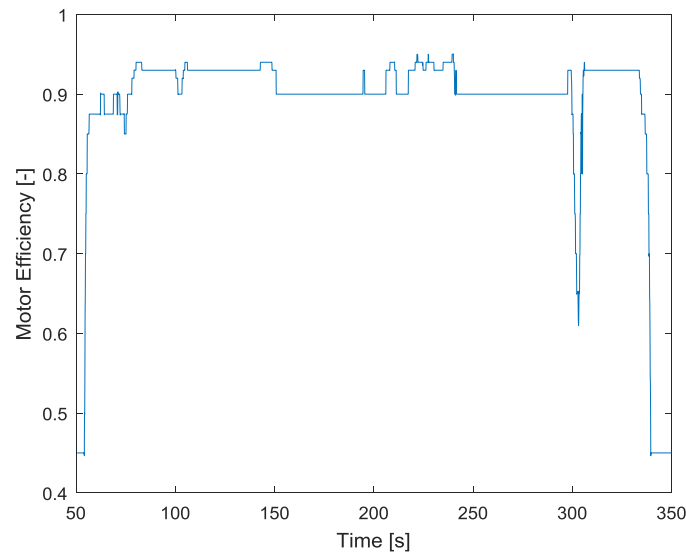


**Figure 5.7** Model of forces acting on the e-bike and resulting tractive force exerted by the e-bike during Route 1.



**Figure 5.8** Calculated motor power draw throughout Route 1.

As previously mentioned, the changes in motor speed and torque which affect the power draw from the battery pack also affect the motor efficiency (Figure 5.9). The peak in the tractive force, resulting in peaks in motor speed and torque around 300 seconds results in a sudden drop in motor efficiency. Due to limitations of the motor efficiency plot (Figure 5.1), whenever the motor speed and torque are greater than manufactures maximum listed values, the efficiency is assumed to be 90%. Additionally, if route 1 were repeated continuously, the model predicts a 12.1 mile range on a battery charge from 100 to 20% SOC. When compared to the simplified to the simplified situation in Chapter 4 which had a range of 20.05 miles, the effects of a real-world situation on the performance of the e-bike show the changes resulting to power draw, motor efficiency, and driving range.

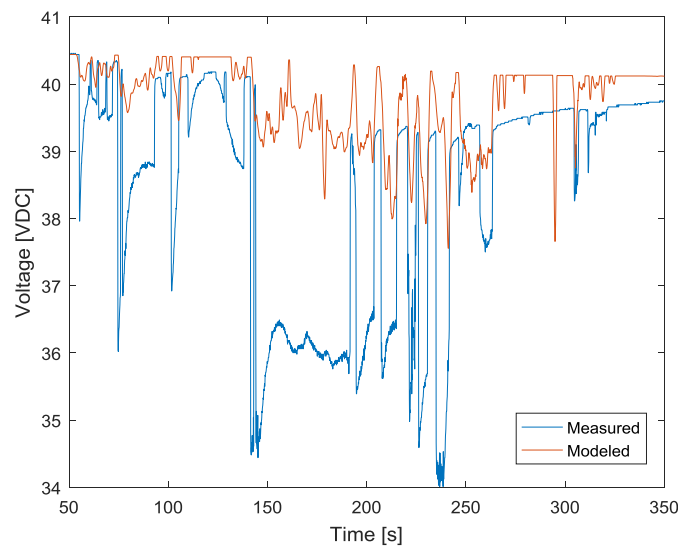


**Figure 5.9** Model of motor efficiency throughout Route 1.

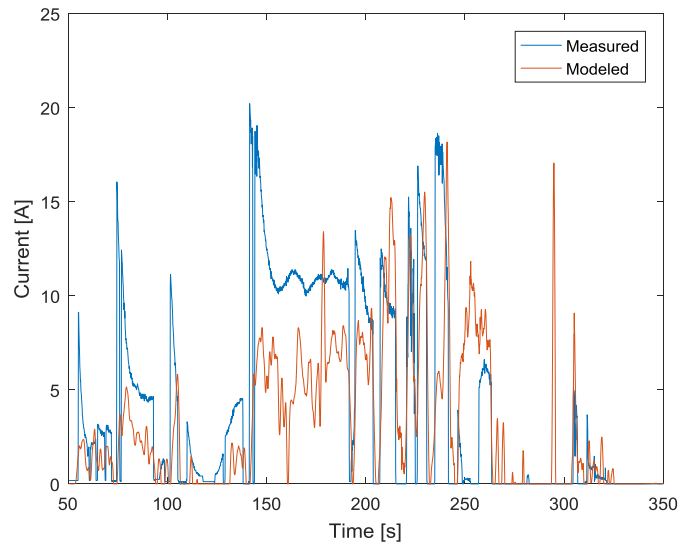
The real test of model validity comes from comparing the resulting battery pack voltage and current draw to the measured voltage and current recorded during testing (Figure 5.10 and Figure 5.11). The model of the battery pack behavior is based on a Hausmann-Decpik model



(Equation 5.2) calibration accomplished for similar NMC cells (Table 5.4) [74]. The e-bike battery pack is configured with 5 parallel series of 10 cells, resulting in a nominal 36 VDC pack with 10 Ah capacity. The voltage for the model is also estimated using this collected data and interpolated as a function of SOC and C-rate. However, the battery model was generated from new NMC cells and, therefore, the model reflects a battery pack operating at peak performance. Considering the age of the battery pack is at least 2 years old and it sat idle for long periods of time, the State of Health (SOH) of the battery pack should be considered judging by the measured voltage data in comparison to the model (Figure 5.10)



**Figure 5.10** Comparison of modeled and measured voltage during Route 1.



**Figure 5.11** Comparison of modeled and measured battery current draw during Route 1.

$$\Delta Q_t = \gamma \left( \frac{I_t}{I_{ref}} \right)^\chi \left( \frac{T_{ref}}{T_t} \right)^\delta$$

**Equation 5.2** Hausmann-Depcik battery capacity model [74].

Hausmann-Depcik Parameter	Calibrated Value
$\gamma$	0.93799
$\chi$	1.03675
$\delta$	0.96661
$I_{ref}$ [A]	1
$T_{ref}$ [K]	298

**Table 5.4** Hausmann-Depcik model calibration for NMC cells.

The SOH of a battery is determined by comparing the maximum available capacity of the cell in its current condition to its nominal capacity when new:

$$SOH = \frac{Q_{usable}}{Q_{nom}}$$

**Equation 5.3** SOH as a fraction of the nominal capacity.

However, there is no model for how SOH affects battery capacity within a single charge or discharge cycle without comprehensive knowledge of the battery pack. Therefore, various modifications were used to attempt to match the battery voltage. The available capacity was found by running through a complete discharge of the battery pack. By raising the rear wheel of the e-bike and turning the throttle to high, the data logger recorded the total current discharge and the time it took to completely deplete the battery pack. When new, the battery pack had a maximum voltage of 42 VDC and a nominal capacity of 10 Ah. Currently, the peak starting voltage is 40.75 VDC and the usable capacity is 5.65 Ah. Using this information, the SOH of the battery pack is 0.565.

When including SOH in the model, the nominal capacity,  $Q_{nom}$ , originally 10 Ah, becomes 5.65 Ah and SOC will become a relative value to better estimate the battery pack's charge; i.e.,  $SOC_r$ :

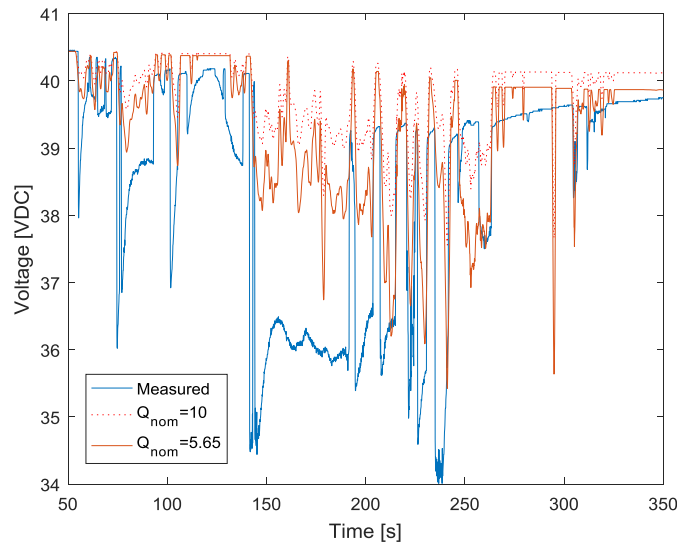
$$SOC_r = \frac{Q_t}{Q_{usable}}$$

**Equation 5.4** Modified SOC for the relative health of the battery pack

As batteries age, they are unable to provide the same voltage range, but will follow similar discharge voltage plots as new cells; however, they are more susceptible to voltage drops with changing C-rates and cell temperature [79]. Therefore, the voltage discharge data used to calibrate the Hausmann-Depcik model remains usable but the SOC and C-rate values should be

adjusted to account for the SOH. As a result, several modifications to the voltage calculations were applied.

First, by changing the nominal capacity to 5.65 Ah, the modeled voltage matched the measured voltage more closely under low current and low C-rate conditions (Figure 5.12). Had the power draw been near constant, it is assumed that the model would more closely match the measured data. However, spikes in power draw during testing caused the battery pack to experience a sudden and larger temporary voltage loss before recovering.



**Figure 5.12** Comparison of the measured battery voltage, the original model and the model using the adjusted nominal battery pack capacity.

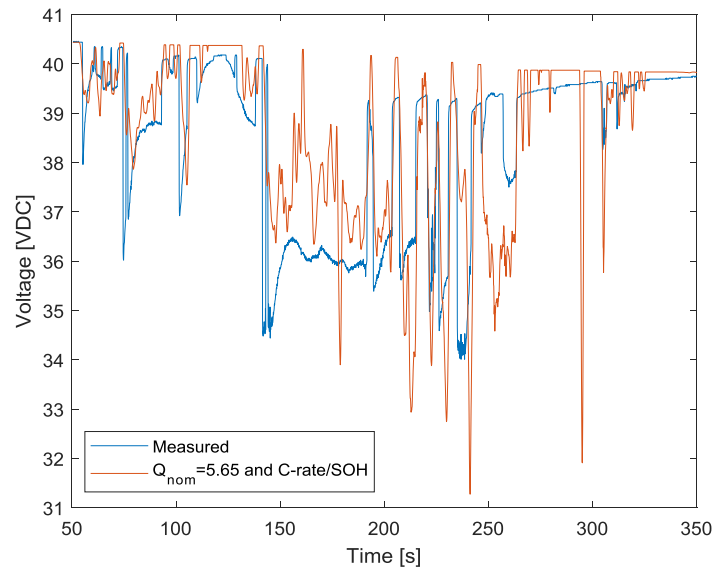
To account for the sharp drops in voltage, the best method is to adjust the C-rate. It is known that operating batteries at higher than recommended C-rates will shorten the lifespan of the cells and cause the SOH to increase faster than at lower C-rates. However, without quantifiable information about how SOH affects C-rate's influence on SOC, the most straightforward method is to include a coefficient factor with the C-rate when estimating the

voltage of the battery pack [80, 81]. Furthermore, as a battery ages, the internal resistance of the cell increases and the nominal capacity decreases [81, 82]. Therefore, according to Ohm's Law, a constant current draw will cause larger voltage drops due to the increasing internal resistance as the cell ages. This implies that as the SOH drops, the coefficient factor for the C-rate will increase. After testing several values for this factor, the best values fell between 1.5 and 2. Probably by coincidence, the inverse of SOH falls within this range and it was used to simplify the model coding:

$$C_{rate} = f \frac{I_t}{Q_{usable}}, \quad f \approx \frac{1}{SOH}$$

**Equation 5.5** Adjustment estimation for C-rate to account for SOH

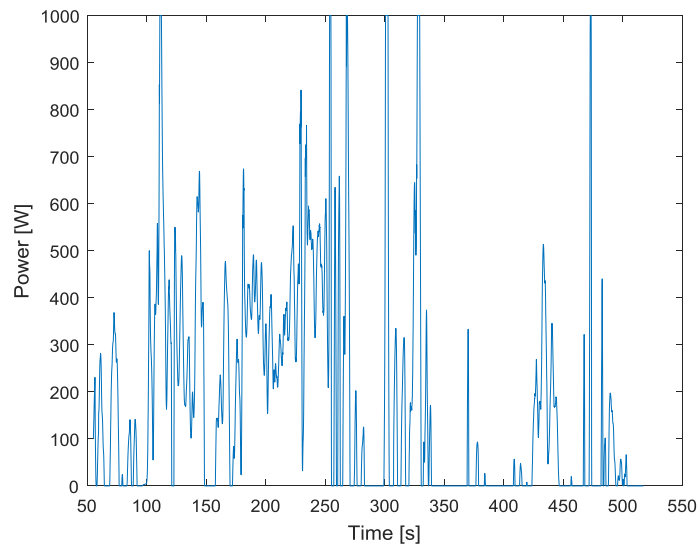
As a result, the modeled voltage data shows more pronounced drops throughout providing a respectively better match over the entire route (Figure 5.13).



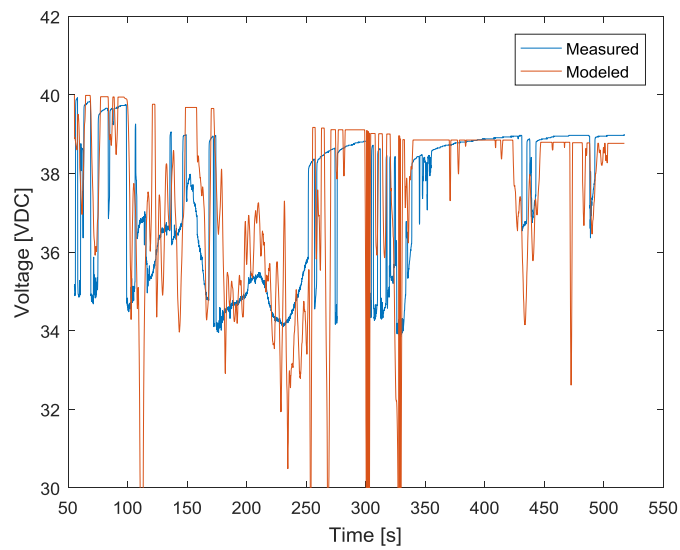
**Figure 5.13** Comparison between measured voltage and model including relative nominal capacity and adjusted C-rate.

One final discrepancy between measured and model data is the effect of the recovery time, best shown around 275 seconds in Figure 5.13. The measured voltage data increases without being charged after the power draw momentarily ends and the measured data remains constant. This is because batteries require time for the voltage to rebound after the current draw ends due to residual chemical reactions occurring inside the battery cell. Additionally, the exact value of the recovery voltage varies depending on the SOC, SOH, and battery temperature [83]. However, the model does not take this voltage recovery time into account and rebounds the voltage immediately between one data sample and the next. To include recovery voltage into the model would require a conditional voltage increase whenever the current draw is momentarily stopped. This voltage increase would be a function of time and a combination of SOC, SOH, or battery temperature and specific to different battery chemistries [83].

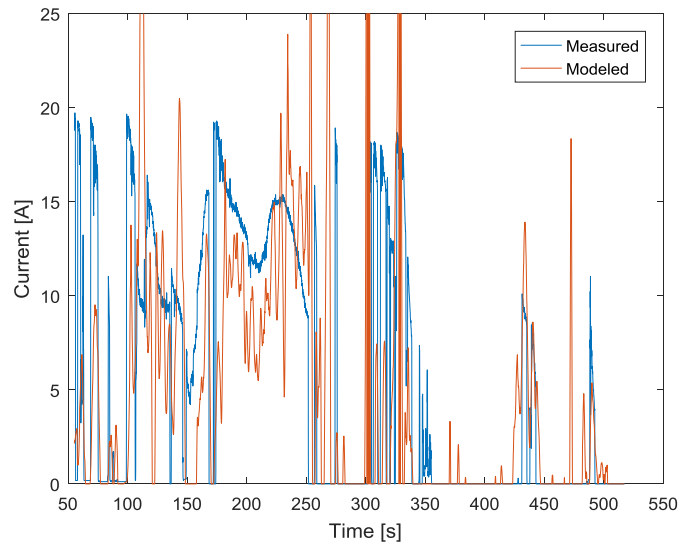
The model up to this point was tested using data collected from Route 1 (Figure 5.2). To ensure that the model is useful, it needs to be validated against other tests. During the test with Route 2, the e-bike accelerated while going uphill resulting in several large calculated motor power spikes (Figure 5.14). Therefore, the only alteration to the model was to limit the maximum motor power possible as listed by the manufacturer [70]. These power spikes caused corresponding increases in current and decreases in pack voltage in the model. However, the overall model behaves respectively similar to the collected data as shown in Figure 5.15 and Figure 5.16.



**Figure 5.14** Motor power draw during testing over Route 2.



**Figure 5.15** Comparison of measured and modeled battery pack voltage over Route 2.



**Figure 5.16** Comparison of measured and modeled current draw from the battery pack over Route 2.

## 5.5 Conclusions

The e-bike model is relatively successful for periods of simple predictability, like near constant speed and power draw scenarios. However, the model struggles to represent collected data during periods of rapid and oscillating change but can show an approximate general trend. Testing to find real values for the coefficient of drag and the rolling resistance coefficient instead of using estimates based on literature values would help fine-tune model results. Furthermore, data interpolated using a representative motor efficiency map and the voltage discharge curves assumes the information is a valid depiction of the direct current motor and battery pack chemistry. However, the model certainty would improve if reference data could be collected directly from the actual motor and battery pack before testing occurs.

The influence of SOH on the behavior of the battery pack is not as straightforward. It is accepted that the battery cell will increase its internal resistance as it ages that will in turn, affect



its ability to supply power. Here, by altering the C-rate through a factor representing the inverse of the SOH, the C-rate would rise as the battery ages helping to approximate the correct trend. However, this might not be an accurate method as applied to other battery chemistries or even other NMC cells. Instead, it simply represents the findings until further extensive research can be done into the influence of aging on C-rate and battery capacity.

Overall, by monitoring the e-bike route, speed, wind, battery temperature, and initial battery voltage while knowing physical specifications about the motor and rider, the model can predict how the battery pack discharges throughout a ride and the voltage of the battery pack at the end. Furthermore, various scenarios will alter the model in predictable ways. A racing cyclist, leaning over the handlebars, will be more aerodynamic and have a smaller coefficient of drag. If regenerative braking can be included in the design of the e-bike, negative current draws could charge the battery pack. Finally, the implementation of a new battery pack would not alter the calculations for the SOC and C-rate.

An individual commuter cyclist, having knowledge about their own bicycle or e-bike and the route they plan to take, could use this model to predict driving range on a full battery charge. Additionally, monitoring the acceleration and net force of the e-bike could alert riders to the distance necessary to stop. If a lidar rangefinder system were also utilized, it would be able to alert commuters to whenever obstacles are within the current stopping distance. By being able to calculate the motor efficiency, the rider could alter behavior to use the motor at its highest efficiency. Furthermore, if the majority of a commuter's e-bike uses the electric motor at a region of low efficiency, the user could replace the motor with one more suitable for their needs.

Route and e-bike modeling would allow commuters to decide if using e-bikes instead of vehicles is suitable for their lifestyles. Additionally, if the general population is made aware of

its capabilities, limitations, and safety features, e-bikes could become more popular while being safety used.

## Appendix A: 2-D lidar system's Arduino microcontroller code

```
#include <Wire.h> //for I2C/TWI communication using SDA and SCL lines

#include <SD.h> //for reading and writing SD cards

#include <Stepper.h> //for controlling stepper motors

const float pi = 3.14159265; //[]

const int phi = 40; //[deg]

const int rpm = 50; //[rev per min]

const int sweepAngle = 100; //[deg]

const float stepAngle = 1.8; //[deg]

const int stepsPerRevolution = 200; //[]

const int chipSelect = 53; //pin

const int leftLED = 40; //pin

const int centerLED = 41; //pin

const int rightLED = 42; //pin

const int carGain = 200; //[mm] = .2[m] (car is 30mph faster than bike)

const int carLength = -4500; //[mm]

int currentDirection = -1; //1 is CW, -1 is CCW

const int criticalDistance = 27000; //27000[mm] = 27[m] ~ 88[ft]

const int leftLaneStart = -2000; //[mm] assuming bike is in center of lane

const int rightLaneStart = 2000; //[mm]

const int leftLaneEnd = -6000; //[mm]

const int rightLaneEnd = 6000; //[mm]
```

```

uint8_t evo[3]; //byte storage, no +/- signs, just number

int sweepCount = 0; //[]

float currentAngle = phi; //[deg]

uint16_t currentDistance; //[mm], no +/- signs, just number

float xDistance; //[mm]

float yDistance; //[mm]

float previouslyDistance; //[mm]

float storedLeftAngle; //[deg]

int storedLeftSweep; //[]

float storedCenterAngle; //[deg]

int storedCenterSweep; //[]

float storedRightAngle; //[deg]

int storedRightSweep; //[]

File dataFile; //create data file

Stepper myStepper(stepsPerRevolution, 8, 9, 11, 12);

void setup()
{
  //Setup Lidar Communication

  #define LidarEvo 0x31 //declare address

  Wire.begin(); //open communication over I2C

```

```

//Setup LED pins

pinMode(leftLED, OUTPUT);

pinMode(centerLED, OUTPUT);

pinMode(rightLED, OUTPUT);

digitalWrite(leftLED, LOW);

digitalWrite(centerLED, LOW);

digitalWrite(rightLED, LOW);

//SD Card Setup

SD.begin(chipSelect);

SD.remove("test.txt"); //delete existing data file

dataFile = SD.open("test.txt", FILE_WRITE); //create blank data file

dataFile.println("Time, Sweep, Angle, Distance, Left, Center, Right");

dataFile.close(); //Move motor to initial position

myStepper.setSpeed(20);

myStepper.step(currentDirection*phi/stepAngle);

delay(100);

}

void loop()

{

//collect lidar distance data

```

```

Wire.beginTransmission(LidarEvo);

Wire.write(0x00);

Wire.endTransmission();

delayMicroseconds(500);

Wire.requestFrom(LidarEvo, 3);

evo[0] = Wire.read(); //First byte

evo[1] = Wire.read(); //Second byte

evo[2] = Wire.read(); //Byte of checksum

currentDistance = (evo[0]<<8) + evo[1]; //[mm]

xDistance = currentDistance*cos(currentAngle*pi/180); //[mm]

yDistance = currentDistance*sin(currentAngle*pi/180); //[mm]

//prepare sd card and data file

String dataString = String(millis()) + "," + String(sweepCount) + "," + String(currentAngle) +
"," + String(currentDistance) + "," + String(digitalRead(leftLED)) + "," +
String(digitalRead(centerLED)) + "," + String(digitalRead(rightLED)) + "\n"; //gather current
data measurements

dataFile = SD.open("test.txt", FILE_WRITE); //open data file

dataFile.println(dataString); //add current data measurements

dataFile.close(); //save and close file

//identify cars and turn on LEDs

```

```

if ((yDistance <= criticalDistance) && (previousyDistance-yDistance <= carGain) &&
(previousyDistance-yDistance >= carLength) && (currentDistance != 1) && (currentDistance !=
0)) //criteria that recognizes a closing in car and no null data
{
if ((xDistance <= leftLaneStart) && (xDistance >= leftLaneEnd)) //only in the left lane
{
digitalWrite(leftLED, HIGH); //turn on LED indicator
storedLeftSweep = sweepCount; //remember the position the car was at
storedLeftAngle = currentAngle;
}
if ((xDistance > leftLaneStart) && (xDistance < rightLaneStart)) //only in the center lane
{
digitalWrite(centerLED, HIGH);
storedCenterSweep = sweepCount;
storedCenterAngle = currentAngle;
}
if ((xDistance >= rightLaneStart) && (xDistance <= rightLaneEnd)) //only in the right lane
{
digitalWrite(rightLED, HIGH);
storedRightSweep = sweepCount;
storedRightAngle = currentAngle;
}
}
}

```

```

//turn off LEDs at same angle on next sweep

if ((digitalRead(leftLED) == HIGH) && (sweepCount == storedLeftSweep+1) &&
(storedLeftAngle == currentAngle))
{
    digitalWrite(leftLED, LOW);
}

if ((digitalRead(centerLED) == HIGH) && (sweepCount == storedCenterSweep+1) &&
(storedCenterAngle == currentAngle))
{
    digitalWrite(centerLED, LOW);
}

if ((digitalRead(rightLED) == HIGH) && (sweepCount == storedRightSweep+1) &&
(storedRightAngle == currentAngle))
{
    digitalWrite(rightLED, LOW);
}

//Change direction as needed

if (currentAngle <= phi)
{
    currentDirection = -1; //CCW

    sweepCount++;
}

```



```
}  
else if (currentAngle >= sweepAngle+phi)  
{  
    currentDirection = 1; //CW  
    sweepCount++;  
}  
  
//turn motor one step  
myStepper.setSpeed(rpm);  
myStepper.step(2*currentDirection);  
  
//Update current angle and distances  
currentAngle = currentAngle-(currentDirection*2*stepAngle);  
previousyDistance = yDistance;  
}
```

## Appendix B: Simple Scenario Model Calculations

$$F_x = m \frac{dv}{dt} = 88.23 * 0 = 0 \text{ N}$$

$$F_D = \frac{1}{2} \rho A_f C_D (v - v_{wind})^2 = \frac{1}{2} * 1.24 * 0.3 * 1.1 * (8.9 - 0)^2 = 13.206 \text{ N}$$

$$F_G = mg \sin \theta = 88.23 * 9.81 * \sin 0 = 0 \text{ N}$$

$$F_R = \mu_r mg = 0.007 * 88.23 * 9.81 = 6.059 \text{ N}$$

$$F_T = F_x + F_D + F_G + F_R = 0 + 13.206 + 0 + 6.059 = 22.265 \text{ N}$$

$$\tau_b = \tau_w = F_T r_t = 22.265 * 0.311 = 6.925 \text{ Nm}$$

$$N = \frac{v}{2\pi r_t} = \frac{8.9}{2\pi * 0.311} = 273.276 \text{ rpm} = 4.555 \frac{rev}{s}$$

$$P_b = 2\pi \tau_b N = 2\pi * 6.925 * 4.555 = 198.160 \text{ W}$$

$$\eta_m = f(N, \tau_b) = 0.90$$

$$P_m = \frac{P_b}{\eta_m} = \frac{198.160}{0.90} = 220.177 \text{ W}$$

## References

- [1] J. Xu, S. L. Murphy, K. D. Kochanek, B. Bastian, and E. Arias, "Deaths: Final Data for 2016," *National Vital Statistics Reports*, vol. 67, no. 5, 2018.
- [2] "Summary of Motor Vehicle Crashes: 2016 Data," National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts. DOT HS 812 580, 2018.
- [3] J. Panter, S. Griffin, A. M. Dalton, and D. Ogilvie, "Patterns and predictors of changes in active commuting over 12 months," *Preventive Medicine*, vol. 57, no. 6, pp. 776-784, 2013.
- [4] J. Panter, S. Griffin, and D. Ogilvie, "Active commuting and perceptions of the route environment: A longitudinal analysis," *Preventive Medicine*, vol. 67, pp. 134-140, 2014.
- [5] B. Gojanovic, J. Welker, K. Iglesias, C. Daucourt, and G. Gremion, "Electric Bicycles as a New Active Transportation Modality to Promote Health," *Medicine and Science in Sports and Exercise*, vol. 43, no. 11, pp. 2204-2210, 2011.
- [6] S. Ji, C. R. Cherry, M. J. Bechle, Y. Wu, and J. D. Marshall, "Electric Vehicles in China: Emissions and Health Impacts," *Environmental Science & Technology*, vol. 46, no. 4, pp. 2018-2024, 2012.
- [7] C. J. Kahane, "Lives saved by vehicle safety technologies and associated Federal Motor Vehicle Safety Standards, 1960 to 2012 - Passenger cars and LTVs - With reviews of 26 FMVSS and the effectiveness of their associated safety technologies in reducing fatalities, injuries, and crashes," Office of Vehicle Safety, NHSTA Technical Report. DOT HS 812 069, 2015.

- [8] "Summary of Motor Vehicle Crashes (Final edition)," National Center for Statistics and Analysis, National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts. DOT HS 812 376, 2020.
- [9] "2016 Fatal Motor Vehicle Crashes: Overview," National Center for Statistics and Analysis, National Highway Traffic Safety Administration Washington, DC, Report. DOT HS 812 456, 2017.
- [10] J. S. Mindell, D. Leslie, and M. Wardlaw, "Exposure-Based, 'Like-for-Like' Assessment of Road Safety by Travel Mode Using Routine Health Data," (in English), *Plos One*, vol. 7, no. 12, 2012.
- [11] "Bicyclists and Other Cyclists: 2016 Data," National Center for Statistics and Analysis, National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts. DOT HS 812 507, 2018.
- [12] *Bicycle Safety* [Online]. Available: <https://www.nhtsa.gov/road-safety/bicycle-safety>.
- [13] *Automated Vehicles for Safety* [Online]. Available: <https://www.nhtsa.gov/technology-innovation/automated-vehicles>.
- [14] J. S. Jermakian, "Crash avoidance potential of four passenger vehicle technologies," *Accident Analysis and Prevention*, vol. 43, no. 3, pp. 732-740, 2011.
- [15] C. D. Mole and R. M. Wilkie, "Looking forward to safer HGVs: The impact of mirrors on driver reaction times," *Accident Analysis and Prevention*, vol. 107, pp. 173-185, 2017.
- [16] A. J. Anarkooli, M. Hosseinpour, and A. Kardar, "Investigation of factors affecting the injury severity of single-vehicle rollover crashes: A random-effects generalized ordered probit model," *Accident Analysis and Prevention*, vol. 106, pp. 399-410, 2017.

- [17] K. Goniewicz, M. Goniewicz, W. Pawlowski, and P. Fiedor, "Road accident rates: strategies and programmes for improving road traffic safety," *European Journal of Trauma and Emergency Surgery*, vol. 42, no. 4, pp. 433-438, 2016.
- [18] (2019). *2017 Infrastructure Report Card* [Online]. Available: <https://www.infrastructurereportcard.org>.
- [19] I. Puente, H. Gonzalez-Jorge, J. Martinez-Sanchez, and P. Arias, "Review of mobile mapping and surveying technologies," *Measurement*, vol. 46, no. 7, pp. 2127-2145, 2013.
- [20] "Lidar 101: An Introduction to Lidar Technology, Data, and Applications," National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, Charleston, SC, 2012, [Online]. Available: <https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf>.
- [21] K. W. Chiang, G. J. Tsai, Y. H. Li, and N. El-Sheimy, "Development of LiDAR-Based UAV System for Environment Reconstruction," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1790-1794, 2017.
- [22] J. Garcia-Gutierrez, L. Goncalves-Seco, and J. C. Riquelme-Santos, "Automatic environmental quality assessment for mixed-land zones using lidar and intelligent techniques," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6805-6813, 2011.
- [23] M. Kelly and S. Di Tommaso, "Mapping forests with Lidar provides flexible, accurate data with many uses," *California Agriculture*, vol. 69, no. 1, pp. 14-20, 2015.
- [24] B. S. Yang, Z. Wei, Q. Q. Li, and J. Li, "Semiautomated Building Facade Footprint Extraction From Mobile LiDAR Point Clouds," *Ieee Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 766-770, 2013.

- [25] R. A. Kromer, D. J. Hutchinson, M. J. Lato, D. Gauthier, and T. Edwards, "Identifying rock slope failure precursors using LiDAR for transportation corridor hazard management," *Engineering Geology*, vol. 195, pp. 93-103, 2015.
- [26] S. R. Neupane and N. G. Gharaibeh, "A heuristics-based method for obtaining road surface type information from mobile lidar for use in network-level infrastructure management," *Measurement*, vol. 131, pp. 664-670, 2019.
- [27] "Lidar Lite v3 Operation Manual and Technical Specifications," Garmin Ltd., 2016, [Online]. Available: [https://static.garmin.com/pumac/LIDAR\\_Lite\\_v3\\_Operation\\_Manual\\_and\\_Technical\\_Specifications.pdf](https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf), Accessed on: 2019.
- [28] I. Blankenau, D. Zolotor, M. Choate, A. Jorns, Q. Homann, and C. Depcik, "Development of a Low-Cost LIDAR System for Bicycles," in *SAE World Congress Experience 2018*, Detroit, MI, 2018: SAE International.
- [29] P. Moore, C. V. Velde, R. Wagner, and C. Depcik, "Design and Analysis of Electric Bikes for Local Commutes," in *ASME 2015 International Mechanical Engineering Congress and Exposition*, Houston, Texas, 2015: University of Kansas.
- [30] "TeraRanger Evo," Terabee, 2018, [Online]. Available: <https://www.terabee.com/wp-content/uploads/2019/03/User-Manual-for-TeraRanger-Evo-single-point-distance-sensors-and-backboards-1-3.pdf>.
- [31] "TeraRanger Evo 60m," Terabee, 2017, [Online]. Available: <https://www.terabee.com/wp-content/uploads/2019/03/TeraRanger-Evo-60m-Specification-sheet.pdf>.

- [32] (2019). *Arduino Mega 2560 Rev3* [Online]. Available: <https://store.arduino.cc/usa/mega-2560-r3>.
- [33] A. Guadalupi, "MEGA2560\_Rev3e," Arduino, Schematic. 2019, [Online]. Available: [https://content.arduino.cc/assets/MEGA2560\\_Rev3e\\_sch.pdf](https://content.arduino.cc/assets/MEGA2560_Rev3e_sch.pdf).
- [34] R. Condit, and Jones, D, PhD., "Stepping Motors Fundamentals," Microchip Technology Inc., 2004, [Online]. Available: <http://www.bristolwatch.com/pdf/stepper.pdf>, Accessed on: 2019.
- [35] "QMot QSH2818 family Manual," Trinamic Motion Control, Manual. 2010, [Online]. Available: [https://www.trinamic.com/\\_scripts/download.php?file=\\_articles%2Fproducts%2Fmotors%2Fqmot-qsh2818%2Fdatasheet%2FQSH2818\\_manual.pdf](https://www.trinamic.com/_scripts/download.php?file=_articles%2Fproducts%2Fmotors%2Fqmot-qsh2818%2Fdatasheet%2FQSH2818_manual.pdf), Accessed on: 2019.
- [36] "Test Results for TeraRanger Evo 60m sensor potential maximum range in varying outdoor conditions," Terabee, 2018, [Online]. Available: <https://www.terabee.com/wp-content/uploads/2019/04/TeraRanger-Evo-60m-Test-Results-Report-Outdoor.pdf>, Accessed on: 2019.
- [37] "Micro SD Card Breakout Board Tutorial," Adafruit Industries, 2019, [Online]. Available: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-micro-sd-breakout-board-card-tutorial.pdf?timestamp=1606065982>, Accessed on: 2019.
- [38] "LP2981-N Micropower 100-mA Ultralow Dropout Regulator in SOT-23 Package," Texas Instruments, 2016, [Online]. Available: <http://www.ti.com/lit/ds/symlink/lp2981-n.pdf>, Accessed on: 2019.
- [39] Babtou67. (2019). *TeraRanger\_Evo\_single\_point\_ArduinoMega\_UART.ino*. [Online]. Available:

- [https://github.com/Terabee/sample\\_codes/tree/master/Arduino/TeraRanger\\_Evo\\_single\\_point\\_ArduinoMega\\_UART](https://github.com/Terabee/sample_codes/tree/master/Arduino/TeraRanger_Evo_single_point_ArduinoMega_UART), Accessed on: 2019.
- [40] *Libraries* [Online]. Available: <https://www.arduino.cc/en/Reference/Libraries>.
- [41] *Language Refrence* [Online]. Available: <https://www.arduino.cc/reference/en/>.
- [42] R. S. Jurecki, Stańczyk, T. L., and Jaśkiewicz, M. J., "Driver's reaction time in a simulated, complex road incident," *Transport*, vol. 32, no. 1, pp. 44-54, 2017.
- [43] "Federal Size Regulations for Commercial Motor Vehicles," U.S. Department of Transportation, FHWA-HOP04-022, 2004, [Online]. Available: [https://ops.fhwa.dot.gov/freight/publications/size\\_regs\\_final\\_rpt/size\\_regs\\_final\\_rpt.pdf](https://ops.fhwa.dot.gov/freight/publications/size_regs_final_rpt/size_regs_final_rpt.pdf), Accessed on: 2019.
- [44] *Smart Fortwo Features and Specs* [Online]. Available: <https://www.caranddriver.com/smart/fortwo/specs>.
- [45] T. Wiklund *et al.*, "Design and Development of a Cost-Effective LIDAR System for Transportation," in *AMSE 2019 International Mechanical Engineering Congress and Exposition*, Salt Lake City, UT, 2019.
- [46] "Mega 2560 Rev3," Arduino, [Online]. Available: [https://content.arduino.cc/assets/Pinout-Mega2560rev3\\_latest.pdf](https://content.arduino.cc/assets/Pinout-Mega2560rev3_latest.pdf).
- [47] B. Earl, "Adafruit Data Logger Shield," Adafruit, 2013, [Online]. Available: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-data-logger-shield.pdf?timestamp=1606065912>, Accessed on: 2019.
- [48] *Arduino Mega Proto Shield Rev3 (PCB)* [Online]. Available: <https://store.arduino.cc/usa/arduino-mega-proto-shield-rev3-pcb>.



- [49] "Stepper Motor 5V 4-Phase 5-Wire & ULN2003 Driver Board for Arduino," Geeetech, 2012, [Online]. Available: <http://eeshop.unl.edu/pdf/Stepper+Driver.pdf>, Accessed on: 2019.
- [50] N. Severson. (2016). *LIDARLite.cpp*. [Online]. Available: [https://github.com/garmin/LIDARLite\\_Arduino\\_Library/blob/master/src/LIDARLite.cpp](https://github.com/garmin/LIDARLite_Arduino_Library/blob/master/src/LIDARLite.cpp), Accessed on: 2019.
- [51] *Arduino Glossary* [Online]. Available: <https://www.arduino.cc/glossary/en/>.
- [52] *Raspberry Pi 4 Tech Specs* [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/?resellerType=home>.
- [53] "Leddar™ Pixell," Leddar Tech, Inc, [Online]. Available: <http://leddartech.com/download/specsheet-leddar-pixell-cocoon-lidar/>, Accessed on: 2019.
- [54] "PX-80™ Handheld Lidar Scanner," Occipital, Inc, [Online]. Available: [https://labs.paracosm.io/hubfs/Gannon/PX-80-Specs%20\(current\).pdf?hsLang=es](https://labs.paracosm.io/hubfs/Gannon/PX-80-Specs%20(current).pdf?hsLang=es), Accessed on: 2019.
- [55] "RedTail LiDAR Systems," 4D Tech Solutions, Inc, [Online]. Available: <https://redtaillidar.com/wp-content/uploads/2020/01/RedTail-Lidar-SpecSheet-October2019-FINAL.pdf>, Accessed on: 2019.
- [56] *RPLIDAR A1* [Online]. Available: <https://www.slamtec.com/en/Lidar/A1>.
- [57] "Scout-16," Phoenix LiDAR Systems, [Online]. Available: [https://www.phoenixlidar.com/wp-content/uploads/2019/12/PLS-SCOUT-16-Spec-Sheet\\_12-19.pdf](https://www.phoenixlidar.com/wp-content/uploads/2019/12/PLS-SCOUT-16-Spec-Sheet_12-19.pdf), Accessed on: 2019.

- [58] *YellowScan Surveyor* [Online]. Available: <https://www.yellowscan-lidar.com/products/surveyor/#contact>.
- [59] C. Depcik, Gaire, A, Gray, J., Hall, Z., et. al., "Electrifying Long-Haul Freight-Part II: Assessment of the Battery Capacity," *SAE International Journal of Commercial Vehicles*, vol. 12, no. 2, 2019.
- [60] *eLogger V4* [Online]. Available: [https://www.eagletreesystems.com/index.php?route=product/product&product\\_id=54](https://www.eagletreesystems.com/index.php?route=product/product&product_id=54).
- [61] P. Mannion, Toparlar, Y., Clifford, E., Hajdukiewicz, M., Andrianne, T., et. al., "On the effects of crosswinds in tandem aerodynamics: An experimental and computational study," *European Journal of Mechanics*, vol. 74, pp. 68-80, 2019.
- [62] R. Beneke, di Prampero, P. E., "Mechanical and metabolic strain of cycling - Analysis with special respect to physiology and biomechanics," (in German), *Deutsche Zeitschrift fur Sportmedizin*, vol. 52, no. 1, pp. 29-32, 2001.
- [63] P. Debraux, Grappe, F., Manolova, A. V., and Bertucci, W., "Aerodynamic drag in cycling: methods of assessment," *Sports Biomechanics*, vol. 10, no. 3, pp. 197-218, 2011.
- [64] F. Grappe, Candau, R., Belli, A., Rouillon, J. D., "Aerodynamic drag in field cycling with special reference to the Obree's position," *Ergonomics*, vol. 40, no. 12, pp. 1299-1311, 1997.
- [65] J. Garcia-Lopez, Rodriguez-Marroyo, J. A., Juneau, C-E., Peleterio, J., et. al., "Reference values and improvement of aerodynamic drag in professional cyclists," *Journal of Sports Sciences*, vol. 26, no. 3, pp. 277-286, 2008.
- [66] W. Hennekam, "The speed of a cyclist," *Physics Education*, vol. 25, no. 3, pp. 141-146, 1990.

- [67] I. Íñiguez-De-La-Torre, Íñiguez, J., "Cycling and wind: does sidewind brake?," *European Journal of Physics*, vol. 27, pp. 71-74, 2006.
- [68] O. Isvan, "Wind speed, wind yaw and the aerodynamic drag acting on a bicycle and rider," *Journal of Science and Cycling*, vol. 4, no. 1, pp. 42-50, 2015.
- [69] J. C. Martin, Milliken, D. L., Cobb, J. E., McFadden, K. L., et. al., "Validation of a Mathematical Model for Road Cycling Power," *Journal of Applied Biomechanics*, vol. 14, no. 3, pp. 276-291, 1998.
- [70] *Heavy-Duty E\_BikeKit No Battery - Rear Wheel* [Online]. Available: <https://www.ebikekit.com/collections/heavy-duty-systems-no-battery/products/heavy-duty-e-bikekit-br-no-battery-rear-wheel>.
- [71] *PowerPhase Pro 100* [Online]. Available: <https://web.archive.org/web/20171011030934/http://evcenters.net/PDF/UQM-PowerPhase%20Pro100.pdf>.
- [72] *Instruction Manual for the eLogger V4* [Online]. Available: <https://www.eagletreesystems.com/Manuals/eLogger%20V4%20Instruction%20Manual.pdf>.
- [73] "Instruction Manual for the Micro GPS Expander V4," Eagle Tree Systems, LLC, vol. 2020 [Online]. Available: <https://www.eagletreesystems.com/Manuals/GPS%20Expander%20Manual.pdf>.
- [74] A. Hausmann, Depcik, C., "Expanding the Peukert equation for battery capacity modeling through inclusion of a temperature dependency," *Journal of Power Sources*, vol. 235, no. 1, pp. 148-158, 2013.

- [75] (2020). *Wind Sensor Rev. P* [Online]. Available:  
<https://moderndevise.com/product/wind-sensor-rev-p/>.
- [76] P. Badger. (2017). *Calibrating The Rev. P Wind Sensor From A New Regression*  
[Online]. Available: <https://moderndevise.com/uncategorized/calibrating-rev-p-wind-sensor-new-regression/?preview=true>.
- [77] *GPS Visualier: Do-It-Yourself Mapping* [Online]. Available:  
<https://www.gpsvisualizer.com/>.
- [78] *Filtering and Smoothing Data* [Online]. Available:  
<https://www.mathworks.com/help/curvefit/smoothing-data.html>.
- [79] V. Marano, Onori, S, Guezennec, Y., Rizzoni, G., Madella, N, "Lithium-ion Batteries Life Estimation for Plug-in Hybrid Electric Vehicles," in *5th IEEE Vehicle Power and Propulsion Conference, 2009*.
- [80] S. Saxena, Xing, Y., Kwon, D., and Pecht, M., "Accelerated degradation model for C-rate loading of lithium-ion batteries," *International Journal of Electrical Power & Energy Systems*, vol. 107, pp. 438-445, 2019.
- [81] D. Wang, Yand, F., Zhao, Yang, and Tsui, K.-L., "Battery remaining useful life prediction at different discharge rates," *Microelectronics Reliability*, vol. 78, pp. 212-219, 2017.
- [82] N. Somakettarin, and Pichetjamoren, A., "Characterization of a Practical-Based Ohmic Series Resistance Model under Life-Cycle Changes for a Lithium-Ion Battery," *Energies*, vol. 12, 2019.

- [83] L. C. Casals, González, A. M. S., García, B. A., and Llorca, J., "PHEV Battery Aging Study Using Voltage Recovery and Internal Resistance From Onboard Data," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4209-4216, 2016.