Dartmouth College

Dartmouth Digital Commons

Dartmouth College Undergraduate Theses | Theses and Dissertations

6-1-2019

Evaluating the Efficacy of Magnetometer-Based Vehicle Sensors

Luke A. Hudspeth
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/senior_theses

Part of the Computer Sciences Commons

Recommended Citation
Hudspeth, Luke A., "Evaluating the Efficacy of Magnetometer-Based Vehicle Sensors" (2019). *Dartmouth College Undergraduate Theses*. 146.
https://digitalcommons.dartmouth.edu/senior_theses/146

This Thesis (Undergraduate) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Undergraduate Theses by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.

Evaluating the Efficacy of Magnetometer-Based Vehicle Sensors

Luke Hudspeth
Dartmouth Computer Science Technical Report TR2019-870
Dartmouth College

Supervisor
Timothy J. Pierson, PhD

In partial fulfillment of the requirements for the degree of
Bachelor of Arts in Computer Science

June, 2019

Acknowledgements

In undertaking my thesis work, I recalled back to my sophomore year when I was just beginning my computer science studies. I was taking COSC 010 (Problem Solving via Object Oriented Programming) with Tim Pierson. At the time, Tim was a PhD candidate (in his forties) and had already had quite the impressive career, and he decided to transition into academia later in life. One of Tim's past ventures entailed working with a sensor company. That company produced a sensor that detects the presence of an automobile in a parking space and could alert drivers to where parking is available in real time. The company had multiple contracts with cities and towns nationwide to implement their system. They were using magnetometers that measured the change in magnetic magnitude to determine the presence or absence of a vehicle. With more modern machine learning, the detection might be optimized and perhaps a more optimal algorithm might be implemented in tandem with these devices. Fast forward to my senior year, I thought back to this lecture two years ago, about a hardware-based parking sensor and went to Tim's office to discuss it. We talked about a potential deep dive into this technology, as well as all the potential areas to improve upon it and potentially create a better methodology using similar magnetometer chips. Tim agreed to serve as my advisor for my senior year thesis work and I would like to extend an enormous amount of gratitude to him for his invaluable guidance, expertise, and help in my culminating experience. Thank you, Tim.

I would also like to thank my family for their support, especially my parents as I called them numerously frustrated with some finicky hardware component, or total system meltdown. Thanks Mom and Dad.

To my thesis committee, I appreciate your consideration for my honors thesis.

-Luke

May 20, 2019

To my Grandfather: Conrad Lynn Darden

Abstract

The issue of parking is more at the forefront of urban development than one might believe. In fact, academic studies have shown that roughly 30% of city traffic is due to drivers circling city blocks attempting to find an open spot. Due to such congestion people often avoid urban centers and downtown areas for shopping or dining because parking is such a hassle and assumed to be unavailable. If drivers knew where parking was available in real time, they could proceed directly to open spaces as opposed to their congestion-inducing attempts to park. A better solution would guide people to available parking and may help re-vitalize downtown areas. The problem of knowing whether or not an available spot exists, however, is complex. This thesis entails an investigation and analysis of the efficacy of magnetometers as vehicle sensors for on-street (non-garage) parking.

While many solutions to detecting available parking have been tried, we focused on magnetometer-based vehicle sensors placed in each parking spot. We built a sensor comprised of a low-cost magnetometer, a radio, a micro-controller, and a battery on a custom printed circuit board. Our idea is that such a sensor could be placed in each parking space and monitor for vehicles. When a vehicle arrives, the magnetometer detects a change in the magnetic environment, then radios the presence of the vehicle in a space to a central server that

aggregates and disseminates parking data to drivers and city officials. City officials could use this data to craft better parking policies and prices. Drivers could then use GPS coordinates and the aggregated space availability in navigation apps to proceed directly to open spaces.

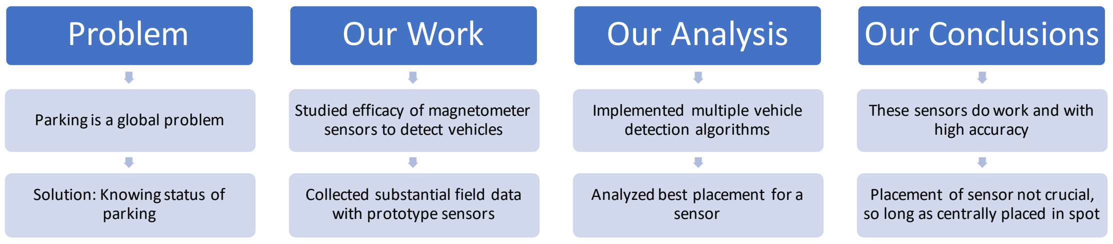
Our hope is that this work will provide a foundation for others to learn from our insights into the reliability, stability, and accuracy of such parking sensors.

After obtaining permission from Dartmouth Parking and Transportation Services, we conducted experiments in a surface lot on Dartmouth College's campus, and as such, we limited our data collection to a single grid-like parking arrangement to gain deeper insight to one common mode of parking. The analysis of the collected data leverages machine learning via sci-kit learn to form a robust detection algorithm for whether or not a vehicle is in a space. We utilized four detection algorithms in total, one via a simple magnitude threshold, another using Gaussian Bayes classification, a decision tree classification model, and finally a random forest model. All of these methods succeeded in correctly detecting the status of a parking spot with accuracy well above 90%. Our best classification model, which uses a decision tree, correctly predicted parking space occupancy with 99% accuracy. In our experiments we show that these sensors are stable and do not drift from their initial reading. Our detection algorithms show that they are an accurate option for vehicle detection. Finally, we show that the placement of a sensor is not crucial, so long as the sensor is centrally placed in a parking spot.

Contents

Chapter 1

Introduction



Problem
Parking is a global problem
Solution: Knowing status of parking
Our Work
Studied efficacy of magnetometer sensors to detect vehicles
Collected substantial field data with prototype sensors
Our Analysis
Implemented multiple vehicle detection algorithms
Analyzed best placement for a sensor
Our Conclusions
These sensors do work and with high accuracy
Placement of sensor not crucial, so long as centrally placed in spot

Figure 1.1: Executive Summary.

1.1 Motivations

To help aid the cities of the future with parking issues, we conducted a holistic study of the effectiveness of magnetometer-based sensors detecting parked vehicles in a surface lot. Our goal is to provide a rigorous foundation that may help smart transportation systems of the near future reduce urban congestion and pollution while increasing downtown utilization of available parking. Such motivation stems from our desire desire to contribute to a more sustainable future as well as aid in a current problem facing all major cities across the world.

1.2 Context of the study

Parking is an everyday chore that can be optimized in a variety of ways. One such area, the area of this study, is the seemingly simple problem of knowing whether or not a car is physically present in a parking spot. Professor Donald Shoup of UCLA has shown that in most urban centers about 30% of all traffic is due to drivers looking for parking [1]. These slow moving drivers, repeatedly circling city blocks near their destination, create increased traffic congestion and pollution on already crowded downtown streets. In another study by the Texas Transportation Institute showed that these inefficiencies cost American drivers alone \$78 billion (4.2 billion lost hours and 2.9 billion gallons of wasted gasoline) [2]. If the location of empty parking spaces were known in real time, drivers (human or autonomous) could simply input their destination into a parking-aware navigation system (envision Google maps) and could be guided directly to the nearest available space, thereby reducing circling-induced congestion and pollution.

Additionally, tracking detailed parking usage data over time can help a city price parking so that supply meets demand. This data-driven pricing could encourage some drivers to select less desirable parking spaces (e.g., around the corner from Starbucks, not right in front) in exchange for a lower price, freeing more desirable spaces. Such a strategy was recently successfully piloted in San Francisco and resulted in improved downtown utilization of parking spaces [3].

To achieve this vision of reduced congestion and pollution with increased downtown utilization of parking, however, parking space occupancy information must be available in real time to assist drivers. One approach to providing this information that has been tested in San Francisco [3], Los Angeles [4], and Washington D.C. [5], is to install a magnetometer-based sensor in each parking space. These sensors detect a change in the Earth's magnetic field caused by the presence of a vehicle in the parking space. Several commercial enterprises have developed these sensors, but data on the sensor's ability to detect different types of vehicles (e.g., sedans, sports cars, or trucks), in different parking scenarios (e.g., parallel parking, head-in, angled), and the detection algorithms used are closely guarded secrets. Furthermore, there is opportunity to explore and perhaps improve the existing technology in terms of cost and accuracy.

1.3 Objectives and Contributions

Our efforts were simpler than developing a city-wide system. The primary objective was to provide a data-driven approach to how these magnetometer-based sensors perform in real-world scenarios. We hope this data can be used as the basis of a future smart transportation system.

Our contributions are as follows: first, we conducted an investigation into the stability and accuracy of magnetometer sensors over time. Second, we conducted field testing to provide clarity on optimal sensor configuration, which included a grid-like pattern of sensors to determine the best placement within a parking space. The third contribution is an analysis of field data with recommendations on where to place sensors in the parking place, as well as the efficacy of various detection algorithms. All of the above information may play a part in developing effective city-wide smart transportation systems.

1.4 Overview of the thesis

We built multiple magnetometer-based sensors to study the optimal placement, configuration, and detection algorithms of various magnetometer-based solutions. Our custom-built prototypes permitted controlling the exact placement of the sensor within a parking space, how often magnetometer readings were collected, and the data from such collection resulted in machine learning-based detection algorithms.

After discussing related work, this thesis is organized as follows:

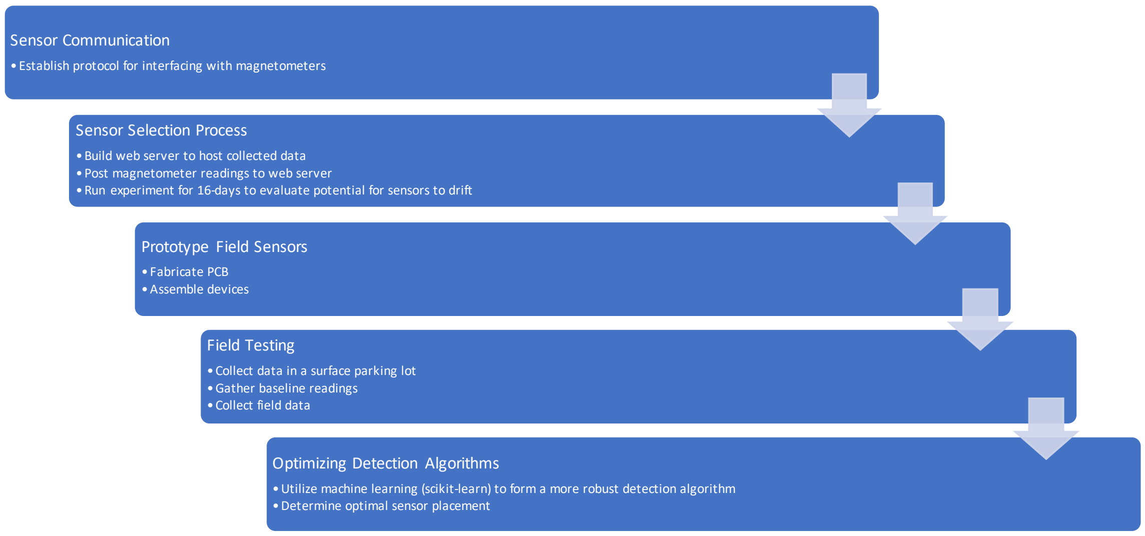

Sensor Communication
• Establish protocol for interfacing with magnetometers
Sensor Selection Process
• Build web server to host collected data
• Post magnetometer readings to web server
• Run experiment for 16-days to evaluate potential for sensors to drift
Prototype Field Sensors
• Fabricate PCB
• Assemble devices
Field Testing
• Collect data in a surface parking lot
• Gather baseline readings
• Collect field data
Optimizing Detection Algorithms
• Utilize machine learning (scikit-learn) to form a more robust detection algorithm
• Determine optimal sensor placement

Figure 1.2: Thesis outline.

Chapter 2

Current Parking Technology and Magnetometer-based Vehicle Sensors

Summary

This chapter highlights the current state of the field in regards to solving parking related issues, particularly the problem of solving the issue of whether or not a vehicle is in a space. First, we broadly explore existing technology, then delve into fixed vehicle sensors, and finally evaluate other magnetometer-based solutions.

2.1 Existing approaches to solving parking problems

2.1.1 Automated garages

Several approaches to solving parking problems have been implemented around the world. One of the most costly solutions entails constructing an automated parking garage shown in Figure 2.1.

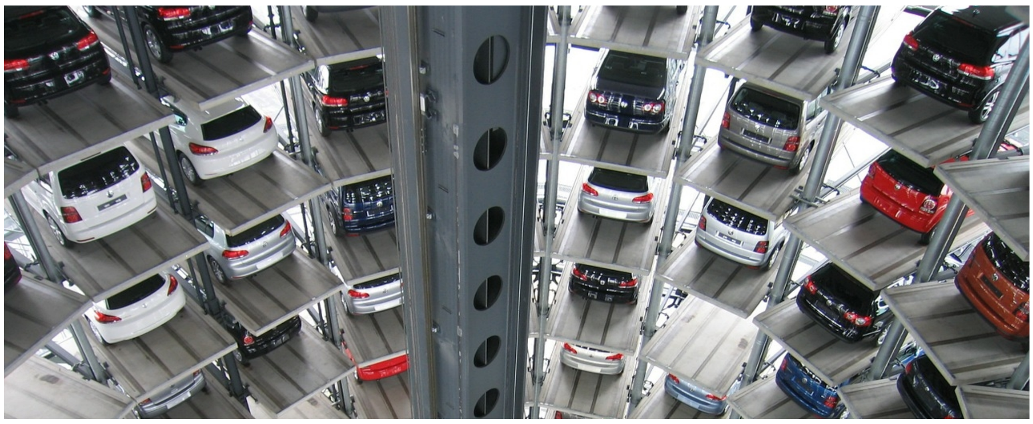


Figure 2.1: Automated parking garage with vehicle elevator [6].

These garages are fully automated and serve as a robotic valet service to users. A car elevator transports a car from street level up into its multi-story garage and parks the vehicle until the customer is ready to retrieve their vehicle. Clearly, these systems are space efficient and impressive, although, the cost for many cities is insurmountable, and constructing these garages is a time-intensive undertaking [7].

2.1.2 Entry and exit garage counters

A fairly simple means of determining garage space availability is through implementing an entry and exit counter (either for the entire garage, or by garage floor). These systems calculate the total number of spaces available in the garage by tracking the number of vehicles that have entered and exited. Such systems work well in garages, however, the counting method is not feasible for on-street parking, which is a much more fluid process without any sort of entrance or exit [7; 8].

2.1.3 Image processing to detect vehicles

Another method of detecting vehicles implements modern image recognition software via cameras mounted either in garages or on streets [9]. There are several disadvantages to this solution, including [7]:

- The cost to implement city-wide cameras is high
- There are privacy concerns around placing so many cameras
- Not an energy efficient solution
- Requires large amounts of computational power and data storage
- There are issues around occlusion (e.g. can't see all the parking spaces due to other vehicles)
- Weather and lighting conditions are variable factors.

2.1.4 Mobile sensors

Researches created a clever system of mobile sensors and inspired a commercial enterprise called ParkNet [10]. They observed that there already exist vehicles which are constantly circling cities such as taxis, city busses, and government vehicles. With circling vehicles "mapping" a city, they then attached ultrasonic sensors to the sides of these vehicles so that when they drive they can detect the occupancy of the spaces. This system operation is shown below in Figure 2.2.

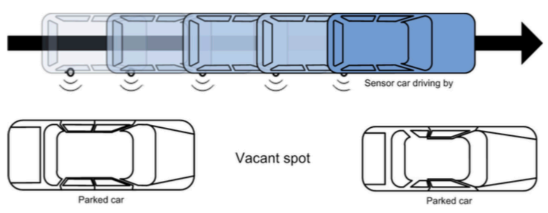


Figure 2: Ultrasonic sensor fitted on the side of a car detects parked cars and vacant spaces.

Figure 2.2: ParkNet ultrasonic system for vehicle detection, which detects vehicles using ultrasonic sound [10].

While the researchers demonstrated that their system functions accurately, there still are several pitfalls to such an implementation. One of which is that there is a low refresh rate. If one of these sensing vehicles does not drive past a block for a long period of time (e.g., 30 minutes) then the parking data becomes stale. Another issue is handling multi-lane roads. If the sensing vehicle is not in the traffic lane directly adjacent to the parking spaces, then the system is unable to detect the state of parking one or two lanes over [10].

2.1.5 RFID tags attached to every vehicle

Another solution involves placing an RFID tag on every vehicle. Thus when a vehicle pulls into a spot, an RFID sensor detects the vehicle's presence [7]. Again there are disadvantages to this solution, primarily an issue of privacy. Many users will likely not want to place such an RFID on their vehicle. These RFID tags could allow an adversary to track the user or otherwise exploit the system. Another issue with this technology, is a nation-wide database of RFID tags would need to be implemented to account for visiting vehicles. Implementing such a nation-wide RFID system is arduous, complicated, and costly [7].

2.2 Fixed Sensors for detecting vehicles

In this section we explore a wide array of fixed place sensing technologies that have been suggested to detect vehicles.

2.2.1 Infrared (IR) sensors

Such sensors detect changes in energy emitted by surroundings and use the change in environment to detect vehicles. IR, however, is highly sensitive to environmental changes (temperature, weather) and does not detect vehicles well outside parking garages [7; 9].

2.2.2 Microwave radar sensors

Microwave radar sensors (Doppler) create a Radio Frequency (RF) field between a sensor and a target and use this zone to detect targets. Although this technology is proven successful in many fields, for parking detection these sensors do not perform well when a vehicle is stopped. Furthermore, because they target any moving object, they can falsely alert easily to non-vehicle signals like a person or other object moving near them [7; 9].

2.2.3 Ultrasonic sensors

Similar to microwave sensors, ultrasonic sensors emit sound waves and calculate the time for the reflected waves to return from a target. Although, these are proven successful in testing, they have several shortcomings. One shortcoming is that these sensors are best implemented in a garage when they can be placed directly over parking spaces. Another disadvantage is that these chips are power hungry. Finally, these sensors have the same risk of falsely alerting to non-vehicle targets that might be temporarily moving over or around a parking space [7; 9].

2.2.4 Light sensors: using visible light backscsatter communication to tag vehicles

A research group has recently implemented a backscatter communication system to track parking availability and vehicle location [11]. The system works by illuminating an area with a light source and backscattering the light with an LCD shutter on a tag. The light source then reads the backscattered light. When the tag is placed in a parking space, the system can detect if a space is empty based on the presence of the reflected light. If the backscattered light is not received by the sensor (because the sensor is occluded by a vehicle), the system declares the space occupied. The tag can also be placed on a vehicle to detect the presence of specific vehicles. This sort of tagging system has the same limitations as RFID tags as discussed in subsection 2.1.5. It also has limitations where the backscatter LCD system could be covered in dirt or water and become ineffective. Additionally, the system has not been shown to work in the presence of bright sunlight.

2.3 Magnetometers and applications

Now we move into the realm of installing fixed magnetometer sensors in every parking space. Magnetometers measure change in magnitude and direction of Earths magnetic field via magnetoresistive technology [12]. Historically, geophysical studies and military applications (detecting submarines) use these sensors.

Recently, however, magnetometers have been widely used in consumer electronics such as cell phones. The disadvantages of magnetometers are that they require close proximity to vehicle to detect a change in the magnetic field [9], and while they are relatively cheap, they are intrusive to implement as they require physical installation in the pavement.

While magnetometers have some limitations, they have been proven viable as vehicle sensors [7]. Our proposed solution centers around placing one such magnetometer-based vehicle sensor in every parking space.

2.3.1 SF Park

We evaluated one Department of Transportation (DOT) project, SF Park, that used this technology to prototype and pilot test a system in the city of San Francisco. That project placed over 8,000 magnetometer sensors over the city and tracked parking availability using their proprietary algorithms. One of SF park's objectives was also to adjust the price of parking to better reflect demand, which we touched on in our introduction. They too saw the advantages of pricing parking elastically so that more coveted spots are priced higher. Drivers then self-select into spaces that meet their needs and spaces that see lower utilization are priced less expensively. SF Park successfully piloted their system in 2013 and in many ways their work served as a launching pad for our investigation and evaluation of this technology [3; 13].

Chapter 3

Sensor Selection

Summary

The primary objective in the initial phases of investigation entailed picking a specific magnetometer sensor for the field tests. To do so, we evaluated the stability of three different magnetometer sensors over time. We want to assess whether these sensors drift from their initial readings or if their readings prove stable over time. Stability is essential for long-term placement of these sensors in the field. For instance, if a large number of sensors were placed in the field, only to later discover that normal readings drifted from their initial magnetic readings, then the sensors would not be stable and future magnetic measurements could be unreliable. If sensor readings drift, constant recalibration would be required to maintain system accuracy. Sensor recalibration is difficult. To recalibrate a sensor, the designated spot must be vacant to get a baseline reading without the presence of a vehicle, which soon becomes costly in terms of time and resources. Therefore, we set out to test the magnetometer's tendency to drift from an initial reading over the span of our field tests, we created a controlled environment free from any variable magnetic influences and measured the sensor's readings every

five minutes for 16 days.

3.1 Experiment design and methodology

3.1.1 Magnetometers evaluated

We considered three different magnetometer chips on breakout boards as candidates for the field sensor. These breakout boards measure the X, Y, and Z components of the magnetic environment, as shown in Figure 3.1.

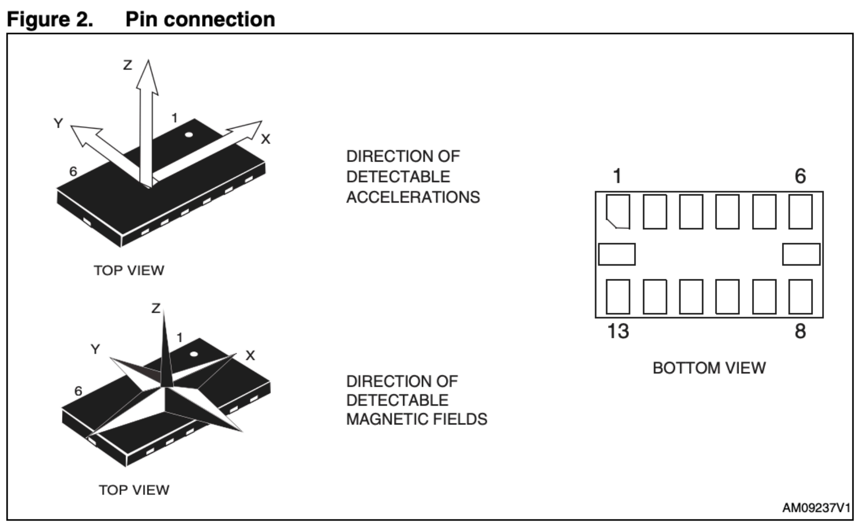


Figure 2. Pin connection
Z
Y
X
1
6
DIRECTION OF DETECTABLE ACCELERATIONS
TOP VIEW
DIRECTION OF DETECTABLE MAGNETIC FIELDS
TOP VIEW
1
6
13
8
BOTTOM VIEW
AM09237V1

Figure 3.1: magnetometer direction of detectable magnetic fields [14].

The specific breakout boards we used were:

1. HMC5883L magnetometer by Honeywell [15] shown in Figure 3.2

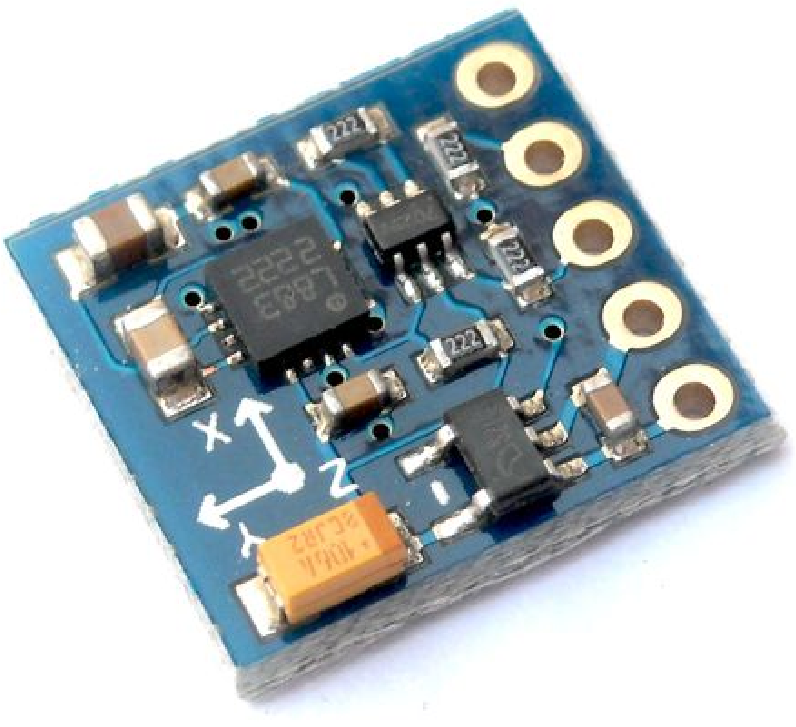


Figure 3.2: HMC5883 magnetometer on a breakout board [15].

.

2. 3-axis magnetometer, gyroscope, accelerometer LSM303DLHC by ST [14] shown in Figure 3.3

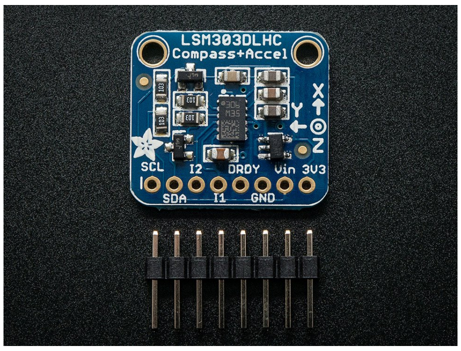


Figure 3.3: LSM303 magnetometer on a breakout board [14].

3. 3-axis magnetometer, gyroscope, accelerometer MPU9250 by Invensense [12] shown in Figure 3.4.

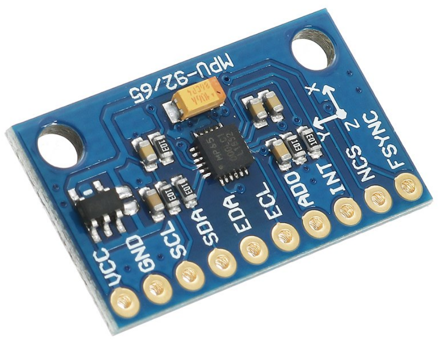


Figure 3.4: MPU92350 magnetometer on a breakout board [12].

3.1.2 Interfacing with the magnetometer chips

The initial system we built was an program (.ino sketch) on an Arduino Uno board [16] that communicated with each of the three magnetometers above. The Uno read each axis of the magnetometer and relayed the magnetometer data directly to the host computer over a USB type 2.0 connection.

3.1.3 Stability test environment

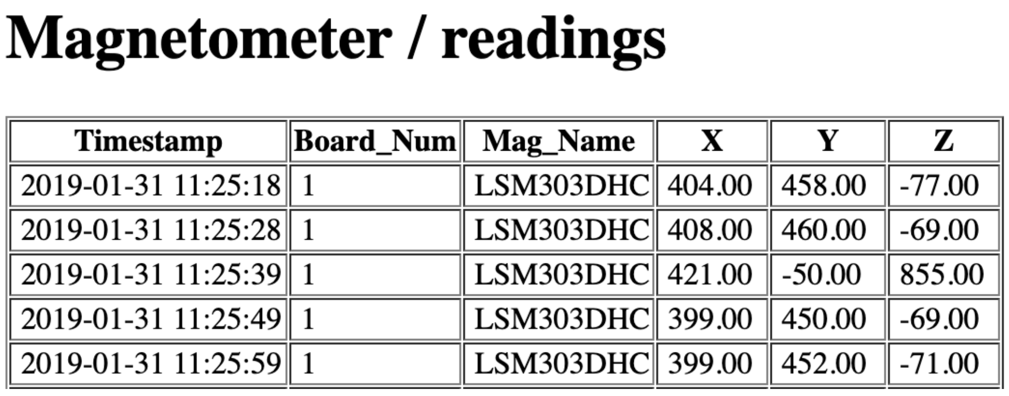
Once we had established communications with each type of magnetometer, we tested them in a location free from variable magnetic influences such as:

1. Moving objects
2. Nearby electromagnetic devices
3. Any chance these sensors might get physically moved
4. High voltage electrical lines.

We placed these sensors underground in a basement where there were no moving objects that would influence the magnetometer readings, with the goal to minimize interference from external variables.

3.1.4 WiFi communications and web server database for stability experiment

To begin our stability experiment, we needed to establish a protocol for interfacing with these sensors remotely. Accordingly, we transitioned from using an Arduino Uno board [16] to using an ESP8266 by Node MCU [17]. The ESP8266 chip has integrated WiFi and runs the code we developed in Section 3.1.2. Using the WiFi component, the sketch we designed connects to a router and pushes the magnetometer readings every five minutes to a web server hosted on a Dartmouth server. An example of the data collected is shown in Figure 3.5.



Magnetometer / readings

Timestamp	Board_Num	Mag_Name	X	Y	Z
2019-01-31 11:25:18	1	LSM303DHC	404.00	458.00	-77.00
2019-01-31 11:25:28	1	LSM303DHC	408.00	460.00	-69.00
2019-01-31 11:25:39	1	LSM303DHC	421.00	-50.00	855.00
2019-01-31 11:25:49	1	LSM303DHC	399.00	450.00	-69.00
2019-01-31 11:25:59	1	LSM303DHC	399.00	452.00	-71.00

Figure 3.5: Server side magnetometer data. We collected magnetometer readings from each breakout board every five minutes for 16 days.

3.1.5 Testing methodology

We designed the architecture for the Arduino devices to collect data for the stability experiment as follows:

1. Connect the ESP8266 to a local WiFi router
2. Take readings from the hard-wired magnetometer
3. Push data every five minutes via the WiFi connection to the web server using a HTTP: PUT function
4. Store that pushed data to our cloud-based Maria database.

3.1.6 Experiment implementation

We tested multiple boards of each sensor type to obtain a clear picture of the stability for each type of magnetometer. We assembled the following seven prototypes using three magnetometer breakout boards as follows:

- LSM303DLHC by ST(three sensors) [14]

- MPU9250 by Invensense (two sensors) [12]
- HMC5883 by Honeywell (two sensors) [15].

Once all seven prototypes were soldered and assembled, we placed them in the test location and ran the experiment for 16 days, sampling the magnetometers every five minutes. For the entire 16-day duration of the experiment, the magnetometers were undisturbed and we collected over 31,000 total readings or roughly 4,500 readings per sensor.

3.2 Sensor selection: results

3.2.1 Stability of sensors over time

We then parsed the data for each sensor reading in the database and calculated the magnitude for each reading via the following formula:

$$M = \sqrt{x^2 + y^2 + z^2}. \tag{3.1}$$

In Figure 3.6 we a plot the magnitude over time; the x-axis is the timestamp, while the y-axis is the magnitude.

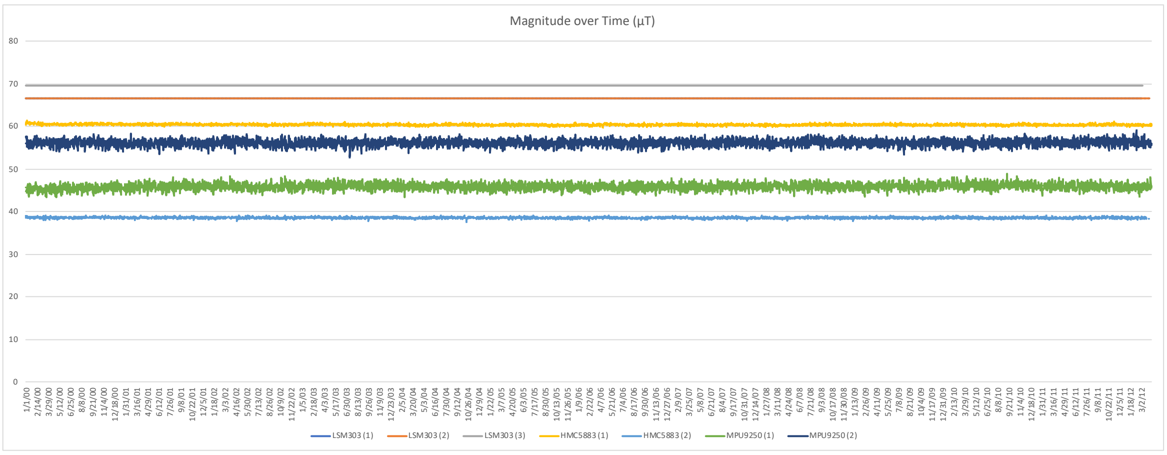


Figure 3.6: Sensor stability results. We plot the magnitude over time to analyze drift, and see that magnitude over time shows slight variations but no drift.

3.2.2 Analysis of drift

The ideal slope for each of these magnetometers would be zero (indicating the sensors do not drift over time). The data exhibits an ideal trend – the slopes of all the regressions are near zero, suggesting that these sensors do not drift away from the initial fixed measurement over time. Table 3.2.2 shows the slopes of a linear regression on the magnitude data.

LSM303(1)	LSM303(2)	LSM303(3)	HMC5883(1)	HMC5883(2)	MPU9250(1)	MPU9250(2)
5.00E-15	5.00E-15	-3.00E-15	-4.00E-05	-9.00E-06	8.00E-05	3.00E-05

Table 3.1: Table: Linear Regression of Magnitude over time shows slopes near zero, indicating little to no drift over duration of experiment.

The data suggests these magnetometers do not drift over the 16 days. In future work, we might examine the stability of the sensors over a longer time period. Although, the 16 days covers the duration of our planned field testing. As the test period is longer than the duration of our planned field testing, this strongly suggests the prototypes will not drift during our collection period.

3.2.3 Standard deviation of sensors

We also calculated the magnitude standard deviation, which we present in Figure 3.7 and Table 3.2.3 .

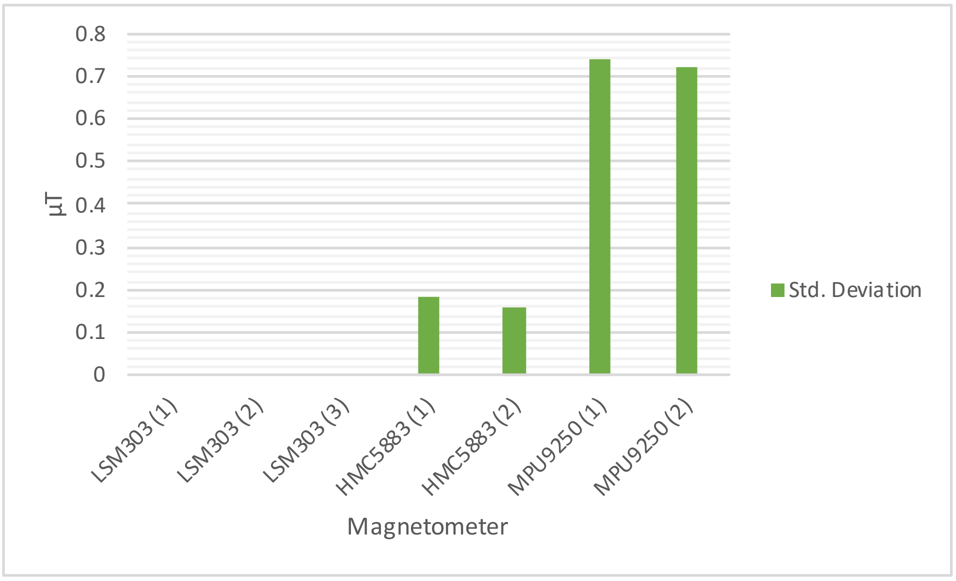


Figure 3.7: Standard deviation of magnitudes. Each sensor appears to have consistent readings.

	LSM303(1)	LSM303(2)	LSM303(3)	HMC5883(1)	HMC5883(2)	MPU9250(1)	MPU9250(2)
Standard Deviation	4.02212E-12	4.00791E-12	2.25978E-12	0.180901458	0.161712436	0.738006401	0.721888628

Table 3.2: Standard deviation of magnitudes over 16 days appears to show consistent magnetometer readings.

3.2.4 Closer analysis shows irregularities

In addition to Figure 3.6 we also plotted the magnitude of each sensor individually. Doing so revealed some concerns with the LSM303 magnetometer. We observe three spikes over the data collection period shown in Figure 3.8 below.

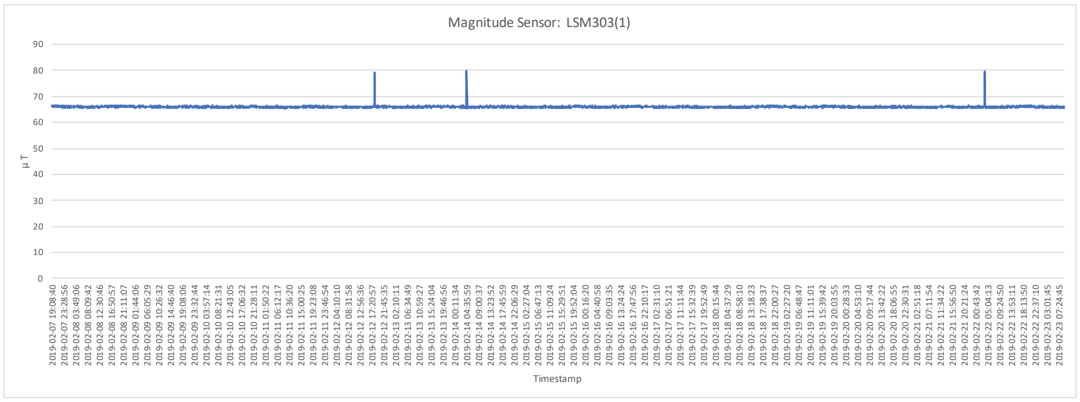


Figure 3.8: LSM303 magnitude over time. The LSM303 magnitude has several spikes in readings.

These irregularities in magnitude concerned us and taking a closer look among the other two magnetometers (the HMC5883 and the MPU9250), and we did not observe these irregularities. We show the same data representation as Figure 3.8 for the MPU9250 in Figure 3.9 and HMC5883 in Figure 3.10.

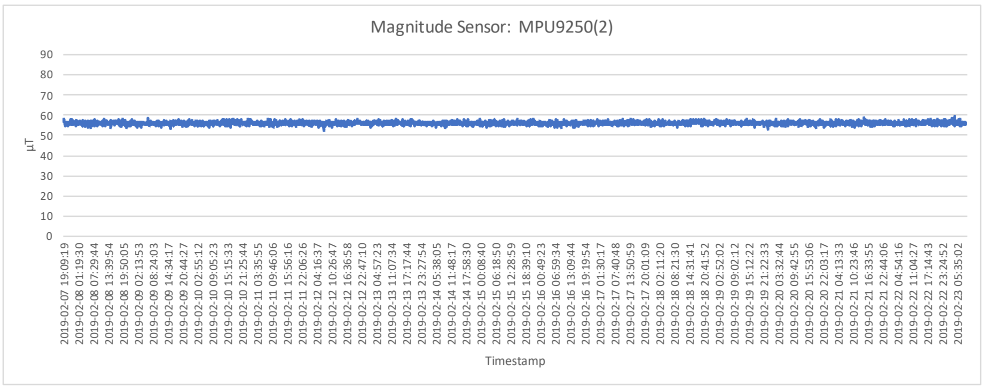


Figure 3.9: MPU9250 magnitude over time. The MPU9250 magnitude did not have the spikes exhibited by the LSM303.

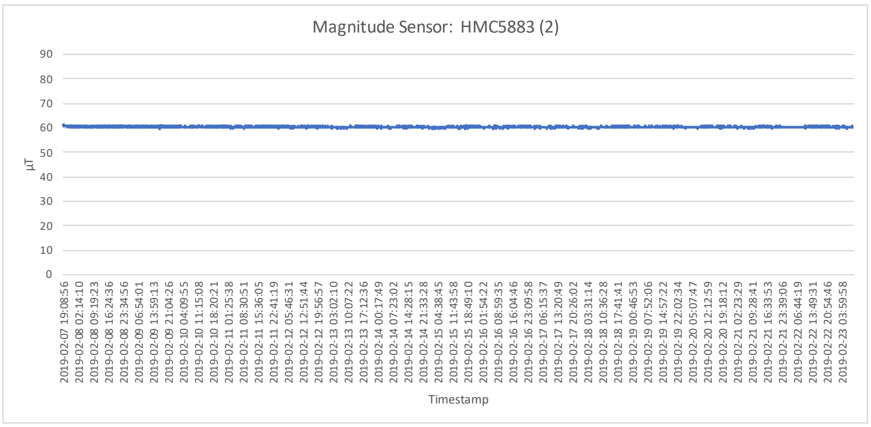


Figure 3.10: HMC5883 magnitude over time. The HMC5833 magnitude did not have the spikes exhibited by the LSM303.

In Figures 3.9 and 3.10, the MPU9250 and HMC5883 magnetometers do not exhibit the same irregularities as the LSM303. While the MPU09250 and HMC5883, do have a higher standard deviation than the LSM303 from Table 3.2.3 the lack of spikes suggests they are more promising candidates for field testing.

3.3 Conclusions

After this portion of the work, we needed to select a sensor to use in the field. We sought a sensor that would not drift and furthermore would not have variability in its readings. From the analysis of each magnetometer's sensitivity and drift behavior, we chose the MPU9250 to serve as the sensor for our prototype going forward. The reasons for this choice are:

- The spikes from the LSM303 concerned us enough to eliminate it as a potential magnetometer for field testing as such spikes would result in instant errors in detection
- We were only ever able to communicate with two HMC5883 magnetometers (we attempted to do so with ten different breakout boards). While the

data for this magnetometer was promising (in terms of lack of drift and low standard deviation), the inability to get 80% of the chips to function eliminated it from our field testing candidates

- The MPU9250 had the highest standard deviation, however, it never demonstrated any of the spikes observed in the LSM303 denoted in Figure 3.8. In addition, all of the MPU9250 chips that we worked with integrated seamlessly with the Arduino programs. Finally, the MPU9250 still had a standard deviation of less than 1 μT and the slopes of the linear regression indicate that it should not drift during field testing (see Table 3.2.2).

Chapter 4

Field sensor prototype and experimental design

Summary

Now we delve into the heart of this thesis work: the field testing and evaluation of this magnetometer-based technology. This chapter discusses the methodology, assembly, and data collection procedures for the magnetometer-based vehicle sensors. The overall objectives for this phase included (1) creating a system to capture the optimal location to place a sensor under a vehicle as well as (2) gathering data to use for training machine learning classification models that we use for detection. Our prototypes were put on a mobile board that we could place under a vehicle. This option was less intrusive than permanently embedding sensors in each space.

4.1 Prototype design and fabrication

4.1.1 NRF24L01 radio communications for field testing sensors

After assessing the stability of the magnetometer sensors (described in section 3.1.1) and choosing the MPU9250 as the designated magnetometer, we designed the vehicle sensors to be used in the field. As WiFi would not be available in the surface lot site, we opted to use radio communication via the NRF24L01 chip [18] between the sensors transmitting and a receiver hosted on a laptop computer. We assembled a two-way channel communication that allowed us to send data over distances up to 200ft, from a sensor placed under a vehicle to the receiver module on the laptop. At this point in the development, we also opted to use a new Arduino board: the Nano [19]. The Nano has much lower power requirements and is a more compact microcontroller than any of the previous models which we used.

This set-up proved more than adequate for field testing as no WiFi routers were required and radio transmission was a simple and power efficient means of running this system.

The radio communication process is shown in Figure 4.1 below.

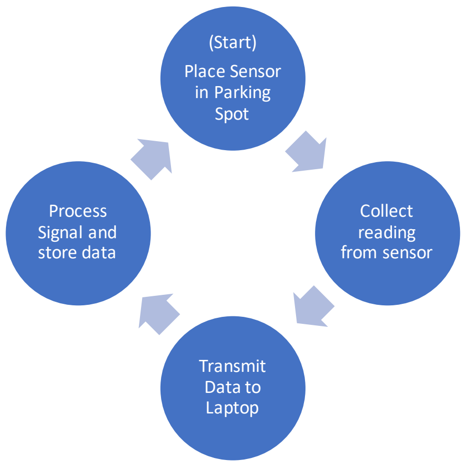

(Start)
Place Sensor in Parking Spot
Collect reading from sensor
Transmit Data to Laptop
Process Signal and store data

Figure 4.1: Radio communications. Sensors placed in parking spot measure magnetic data and transmit the data over an NRF24L01 radio to a laptop for analysis.

4.1.2 Radio communications

For our implementation, the transmitters are constantly transmitting their data, however, the receiver only records data when the user specifies. This enabled us to precisely set up the system for measurement before taking any readings (see Section 4.3 for more details).

4.1.3 PCB fabrication using JLCPCB and Fritzing

We built the full system on breadboards to ensure functionality, however, breadboards would not be suitable for the field as there is the potential for connections to come loose, the magnetometer to move, and overall it is not as durable as a printed circuit board (PCB). Therefore, we fabricated custom printed circuit boards. Each PCB had three electrical components, the Arduino Nano microcontroller, the NRF24L01 radio, and finally the MPU9250 magnetometer. Using JLCPCB [20], a PCB board fabricator, we designed the desired board using Fritzing [21] then sent JLCPCB a .gbr (gerber) file, which JLCPCB fabricated. The images of the pre-fabricated board and an assembled prototype are shown in Figures 4.2 and 4.3. We soldered all the components ourselves.

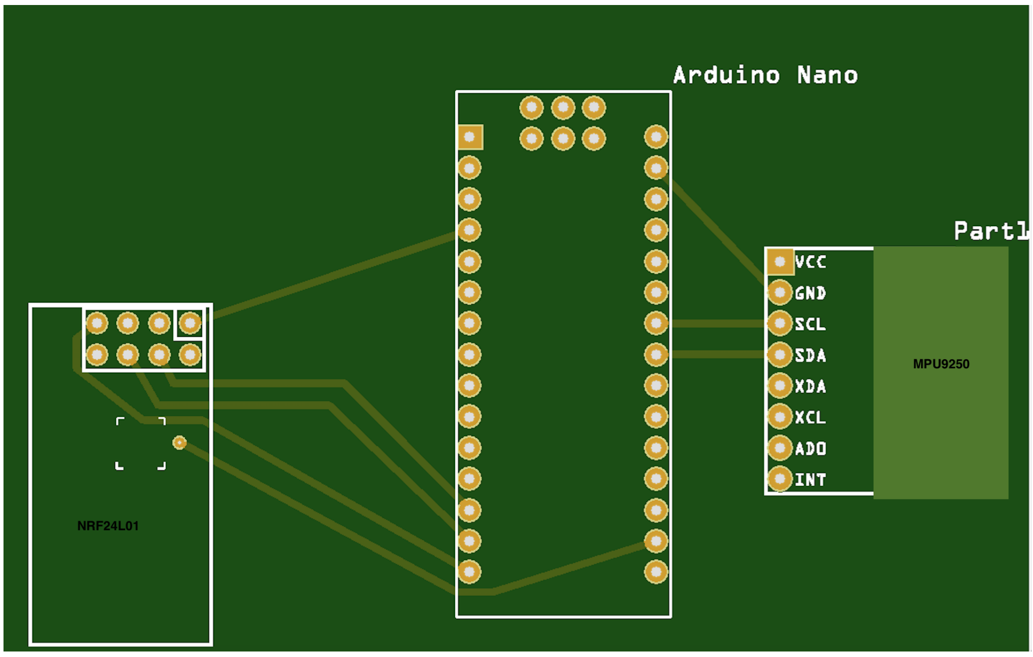


Figure 4.2: We used Fritzing, a CAD software, to design our system. This image is the output of that design for the pre-fabricated PCB.

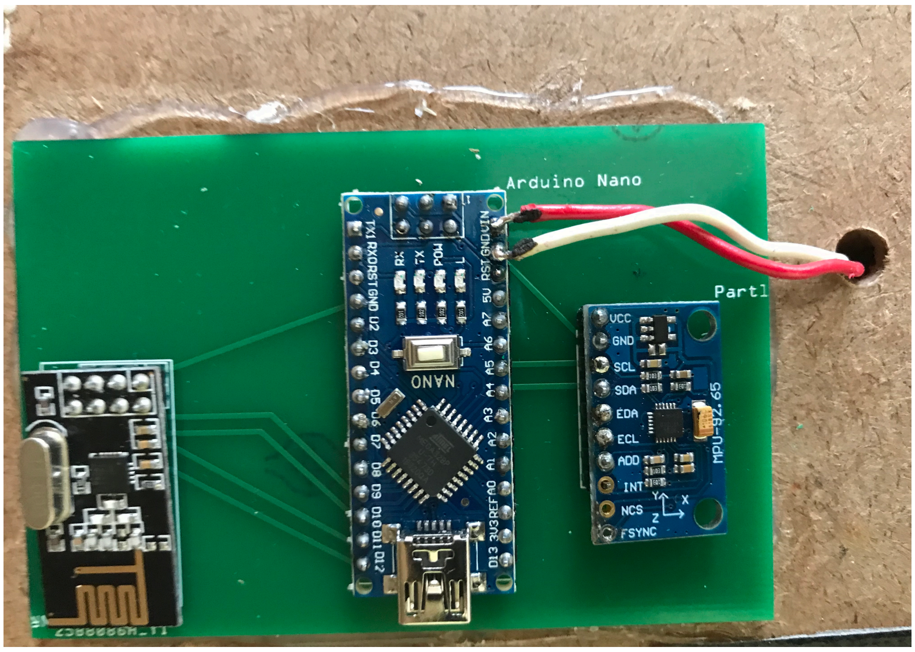


Figure 4.3: An assembled prototype that we used for field testing.

4.1.4 Prototype sensor board fabrication

One objective in the field testing phase is to determine the best location under a vehicle to place a sensor. i.e., directly under the engine bay, under the vehicle chassis, or more centrally located under the vehicle. To capture this data, we chose to implement nine sensors in a grid-array mounted on a board covering a 2' by 4' area. Such a spread permitted gathering data across a wider surface area under the vehicle than if we were to simply use one sensor. Furthermore, using nine sensors also led us to collect nine times as many data points to later use for training our machine learning models. We used a laser cutter to cut out the places for the sensors to ensure the sensors would be equally spaced and to eliminate human error in fabricating the placement on the sensor board. After cutting out the designated spots for the sensors, we soldered and assembled the PCB boards and components.

We assembled nine field prototypes (with a few spares in the event of sensor failure) and secured them to the cut out board. An issue we foresaw in powering these devices is changing out the batteries that would provide power. If the batteries were within the sensor box, then replacing a battery might shift the sensors and skew later readings. Therefore we opted to wire a "power box" of 9V batteries externally from the central box. We then used corrugated plastic over insulated wiring (all under the sensors) which then feed up through the board and connect to the Vin and Ground pins. The wires lead out to the external power batteries eight feet away. The fabrication and assembly process is documented in Figures 4.4-4.6 below.

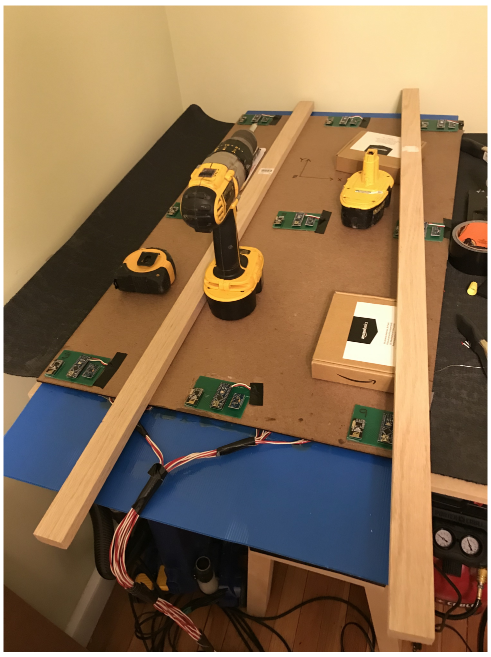


Figure 4.4: Here we are securing the laser cut board and sensors to the corrugated plastic.

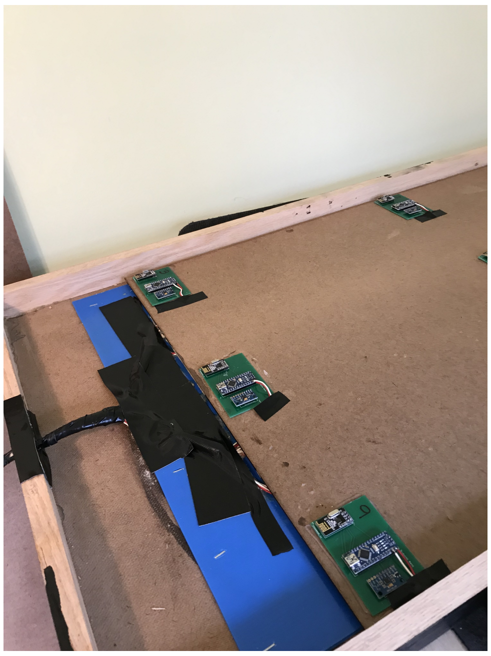


Figure 4.5: We run all the wiring for powering the sensors outside of the sensor box to the batteries.

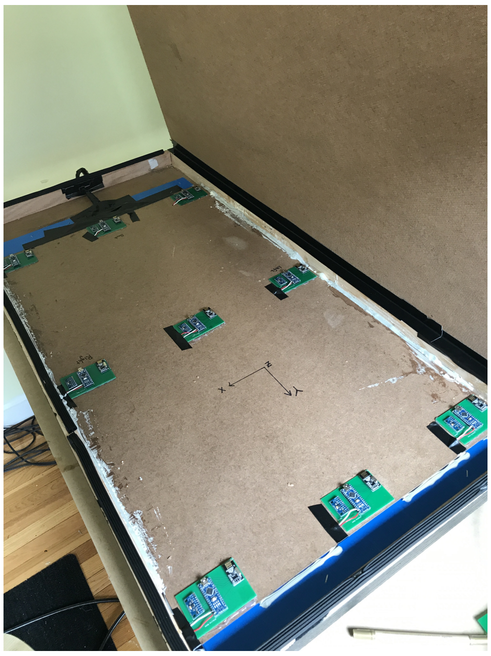


Figure 4.6: The finished product, fully assembled and insulated.

4.2 Surface lot selection and preparation

Our next task to begin data collection was to choose the site at which we would collect data. We sought to choose a site that met the following characteristics:

- Free from potential interfering signals (electrical lines or construction)
- Representative of a real-world parking site
- High turnover rate to collect a variety of different vehicle arrangements (both in the space and in neighboring spaces).

After compiling these desired features and receiving permission from Dartmouth's Parking and Transportation services, we settled upon Dewey Parking lot: G Lot as the best site for the data collection. We chose 10 spots from this particular lot that formed a grid pattern represented in Figure 4.7 below.

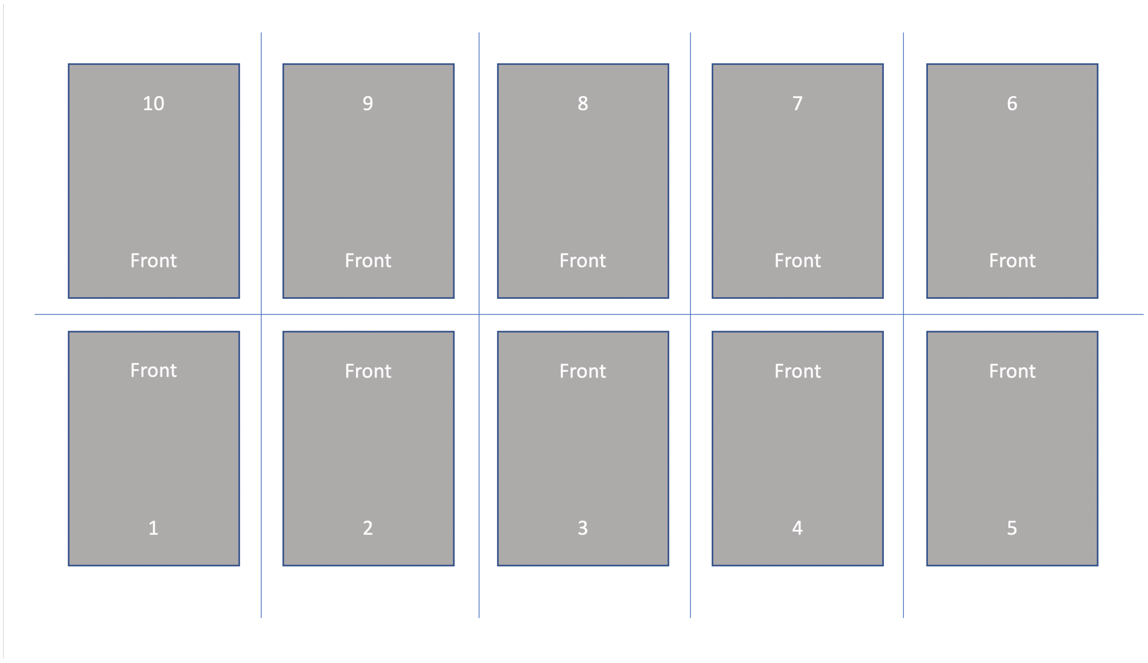


Figure 4.7: Surface lot parking arrangement; in this arrangement vehicles parked head-to-head.

We sought out this particular arrangement, as this grid is clearly a highly used real-world parking lot (examples include surface lots everywhere i.e., grocery stores, airports, shopping malls, etc.). Another advantage of this grid is the simplicity, we did not choose to handle edge cases like parallel parking or garage parking that might be more complex; alternative parking scenarios were out of the scope of our evaluation.

Our procedure was to place a board with nine sensors under a vehicle, take readings, then move the sensor board. We then spray painted and marked the designated spot for the sensors to be placed. We sought to avoid variability in placing the sensors so the spray painted marks served as a means of placing the sensors in the same position. Finally, we numbered the spots as depicted in the Figure 4.7, so we could track the spot during our data collection, (for later subtraction of initial baseline readings).

4.3 Data collection suite procedures and functionality

Using Python, we built both a GUI and a software collection program to interpret serial data from transmitters and store data into a .csv file. The program collects readings from all 9 sensors then appends a data object to the .csv file.

We track key features about the parking environment. For example, we capture the type of vehicle occupying a space (e.g. vacant, sedan, truck, SUV, motorcycle). As well as:

- Spot number
- Timestamp
- Parking arrangement of neighboring spots (including vehicle type)

- x, y, and z values for sensors 1-9.

We sought to capture as many data points as possible when in the field, to ensure that the later machine learning and data analysis would have as much data as possible. The surrounding spots are shown in Figure 4.8.

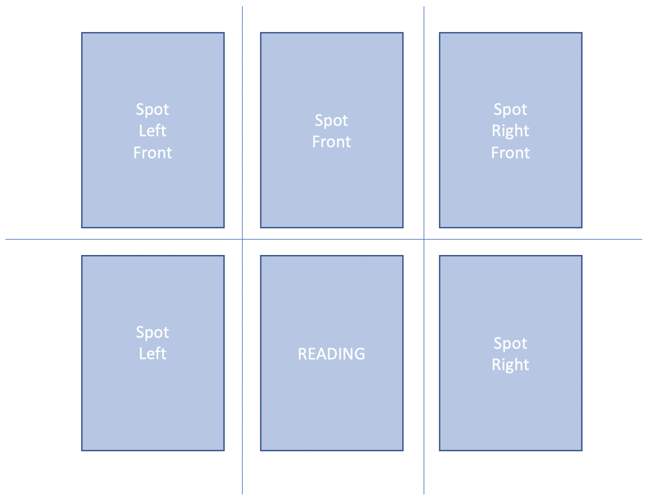


Figure 4.8: Data capture. We collected occupancy data on the spot in question as well as the surrounding spaces.

The software side implementation GUI appears in Figure 4.9.

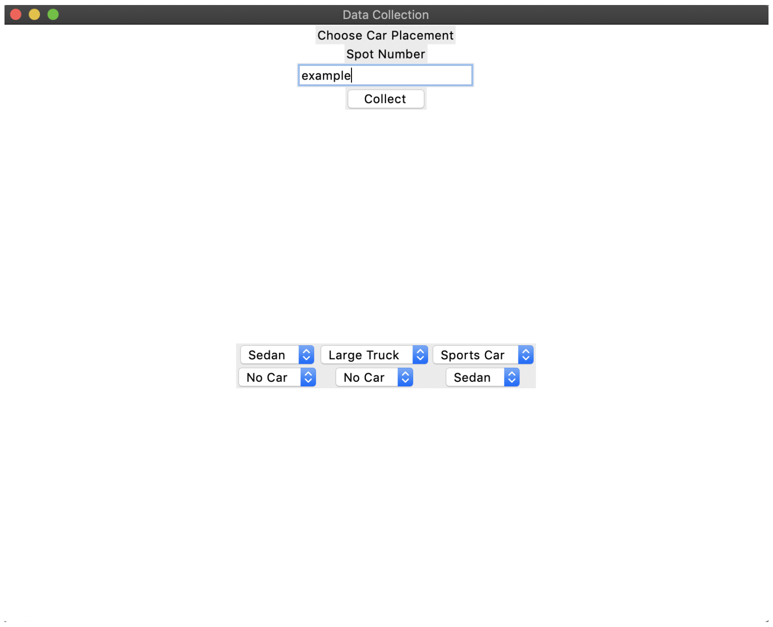


Figure 4.9: The GUI view for our data collection suite, which we used to capture all of our field data and baseline readings.

Whenever we press "collect," a data object is created that parses the x,y and z magnetometer readings as strings, iterates through each of the 9 sensors in order, then stores all the data as a row of a .csv file name labeled for that day of data collection i.e. "data_05_04.csv".

4.4 Baseline data collection and cross check

Another necessary component is the initial calibration or collection of baseline readings for each parking space. This step is essential to subtract out the initial readings of an empty spot to "equalize" each data collection for a spot; the magnetic readings for empty spaces will be different by space. We went to the surface lot site on a day during which no cars whatsoever were present and gathered these baseline readings for spots 1-10. See Figure 4.10.

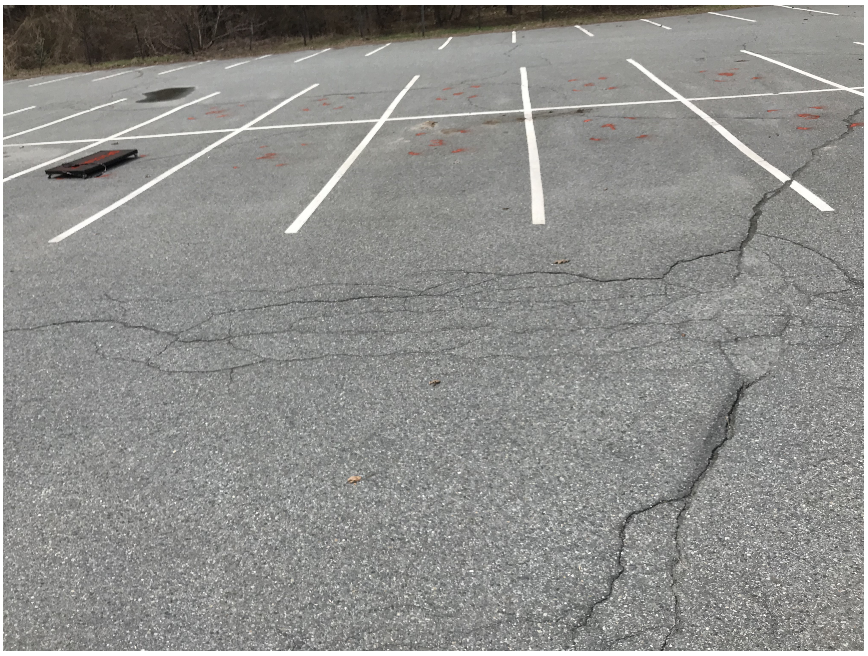


Figure 4.10: Collection of baseline readings. The sensor board can be seen on the left side of the picture as well as the spray painted marks denoting the space number and the sensor board placement.

4.5 Data collection procedures

To limit variability and maximize consistency, we followed a checklist each time we collected data:

1. Label date of data collection
2. Turn on power to sensors 1-9
3. Run Data_Collection.py
4. Place prototype board in the spray painted outline (under a vehicle or if spot is empty)
5. Select the accurate parking arrangement in the software GUI

6. Press collect and wait for Sensors 1-9 to relay their readings

7. Move board to next designated spot and repeat steps 4-6

8. Terminate data_collection.py and turn off power to all sensors.

Over the course of the field testing, we collected approximately 1,100 data points (for each sensor thus totalling approximately 9,900 total readings). Included below are Figures 4.11 – 4.14 documenting field testing.

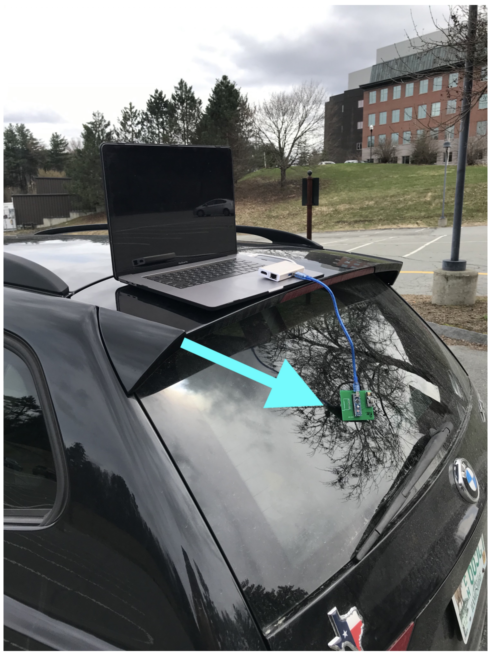


Figure 4.11: Here we are collecting field data, observe the laptop and receiver module (green PCB).

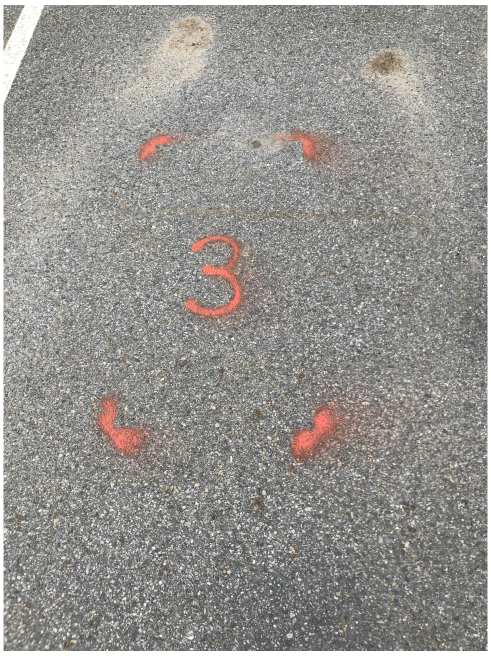


Figure 4.12: Close up of spot #3. We marked four corners to designate the placement of the sensor board. This ensured the sensor board was placed in the same location for each reading.

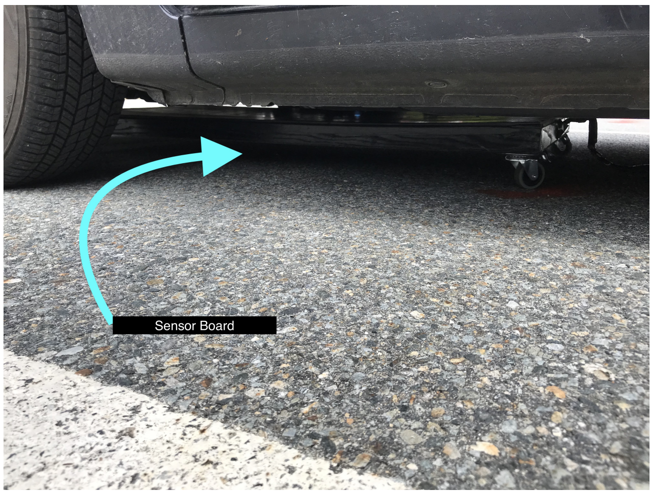


Figure 4.13: We placed the sensor board under vehicles, lining up the board with the spray painted markings.

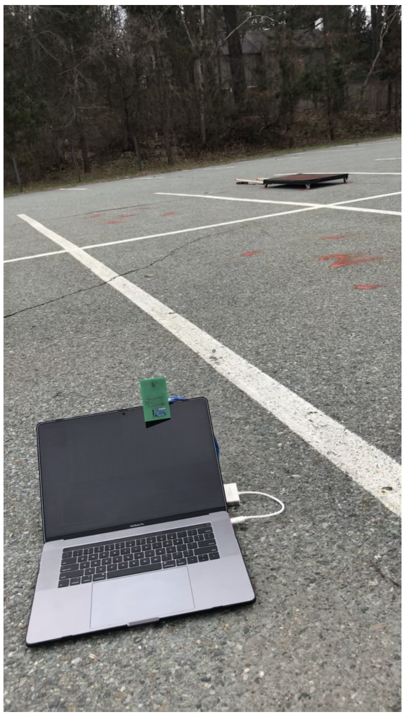


Figure 4.14: We collected all the baseline readings when no vehicles were present.

Chapter 5

Experimental results field testing: detection algorithms and optimal sensor placement

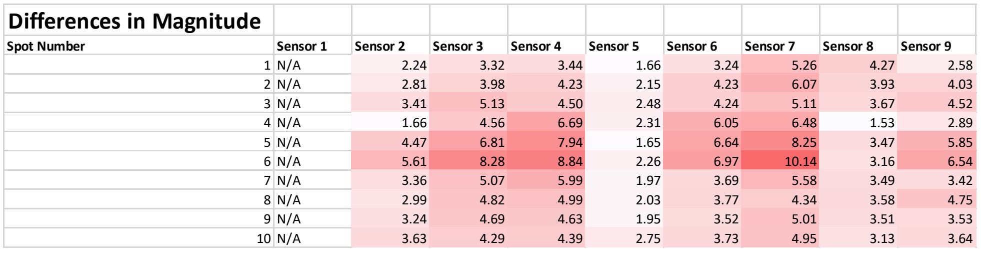
Summary

This chapter is the culmination of the thesis work; the analysis of all the field data collected over several weeks. The analysis provides clarity on the method in which the data is initially processed, elucidates the functionality of the machine learning and detection algorithms, and finally shows the success of this honors thesis work. Each section delves in to some particular component of the final data analysis.

5.1 Baseline readings and field data processing

5.1.1 Baseline readings

To make parking space occupancy predictions, we took a crucial first step: the calculation of initial baseline value averages and then subtraction of those averages from the actual sensor readings. This step accounts for the magnetic variability between parking spaces (e.g., one space may be near an underground power line, while other spaces have no major permanent magnetic influences). By subtracting out the baseline, we "equalize" or calibrate the field readings against the baseline (in which no cars are present). In this baseline phase of data collection we visited the surface lot site twice. We took the differences of these two baseline data collections (expecting them to be zero). The results of those two visits and their differences are in the Table 5.1.



Differences in Magnitude

Spot Number	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8	Sensor 9
1	N/A	2.24	3.32	3.44	1.66	3.24	5.26	4.27	2.58
2	N/A	2.81	3.98	4.23	2.15	4.23	6.07	3.93	4.03
3	N/A	3.41	5.13	4.50	2.48	4.24	5.11	3.67	4.52
4	N/A	1.66	4.56	6.69	2.31	6.05	6.48	1.53	2.89
5	N/A	4.47	6.81	7.94	1.65	6.64	8.25	3.47	5.85
6	N/A	5.61	8.28	8.84	2.26	6.97	10.14	3.16	6.54
7	N/A	3.36	5.07	5.99	1.97	3.69	5.58	3.49	3.42
8	N/A	2.99	4.82	4.99	2.03	3.77	4.34	3.58	4.75
9	N/A	3.24	4.69	4.63	1.95	3.52	5.01	3.51	3.53
10	N/A	3.63	4.29	4.39	2.75	3.73	4.95	3.13	3.64

Figure 5.1: Differences in two baseline data collections. This table shows the difference in magnetometer readings in each parking space from two samples taken when no vehicles were present. We see the differences are near zero.

In Table 5.1 we observe the greatest difference in any axis is 10.14μT, although most readings are near zero. When a vehicle is present, each axis normally differs by more than $40\mu\text{T} - 100\mu\text{T}$ from its baseline reading. Future work would explore fixing sensors permanently to avoid variable change between baseline readings[1].

[1]Note that sensor 1 is not included in these differences as sensor 1 required replacement due to hardware failure before actual field data collection began.

5.1.2 Field data processing

After baseline collection, we then began field data collection, following the outline from Section 4.5. For processing field data, we averaged the baselines we collected from two visits (although excluding sensor 1 as it only had one baseline session) and then subtracted these average baselines from the magnetometer data collected by the sensors. We separated the data out by sensor (1-9) and began the modeling of detection algorithms. We collected 9,882 readings for the model training and evaluation.

5.2 Detection algorithm: magnitude

First we evaluated using the magnitude of the X, Y,and Z readings to detect vehicles. This simple magnitude-based vehicle detection algorithm uses a pre-defined threshold. If the magnitude of the magnetometer's readings exceeds the threshold, then the algorithm indicates a vehicle is present in the parking space. If the magnitude of the reading is below the threshold, then the algorithm indicates the space in empty. We used the following formula to calculate magnitude:

$$M = \sqrt{(X)^2 + (Y)^2 + (Z)^2}.$$

We used a data table to plot the different accuracy achieved by changing the threshold value e.g., if magnitude exceeds 13.0 μT then indicate this space is occupied, otherwise indicate the space is vacant. The accuracy plot using this formula is shown in Figure 5.2 and the numeric outputs are shown in Figure 5.3. In Figure 5.3 we observe that the optimal magnitude threshold is 13.0 μT with an accuracy of approximately 92.5%.

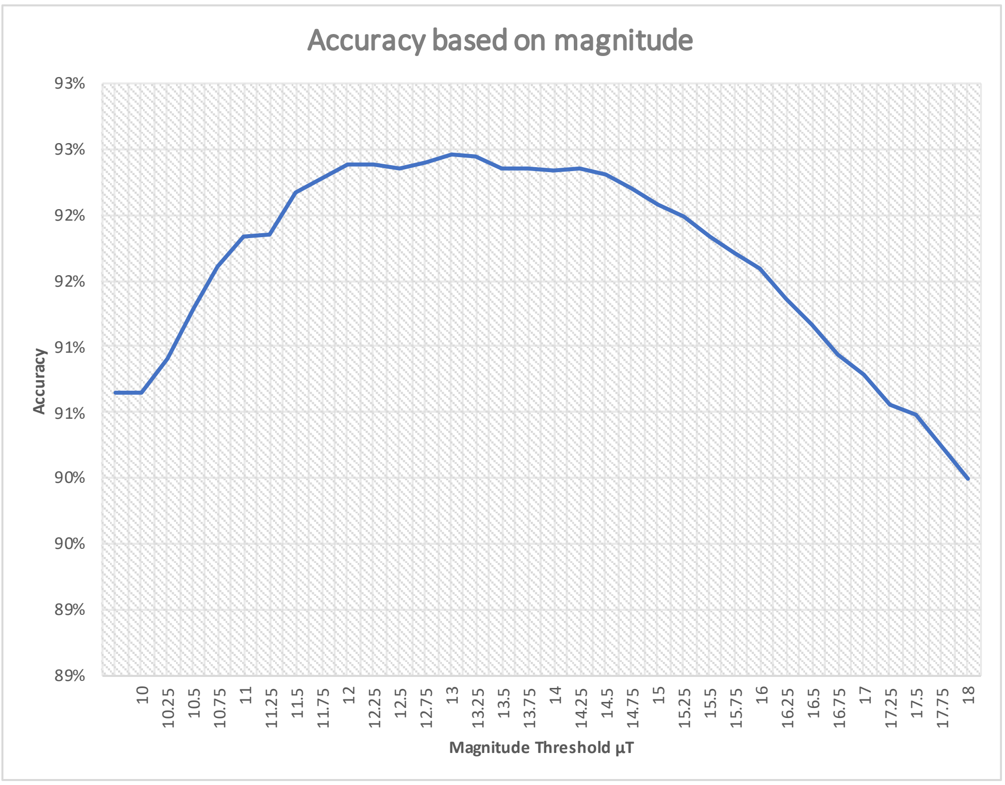
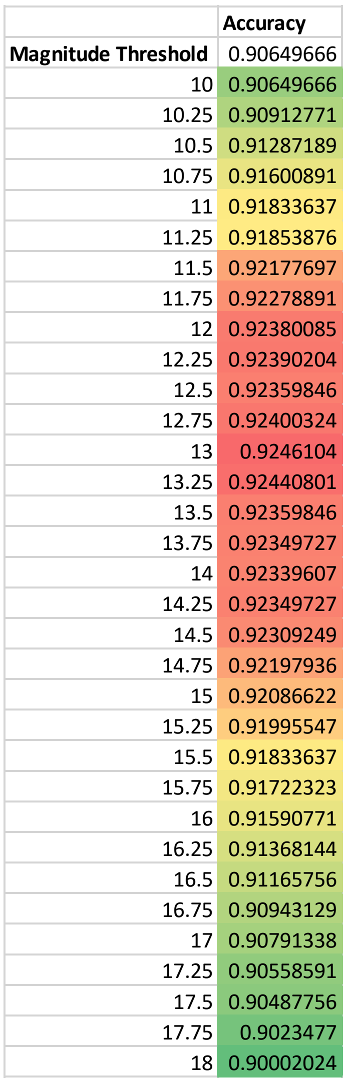


Figure 5.2: Accuracy at different magnitude thresholds. We see the highest accuracy between 12 and 14 μT.



	Accuracy
Magnitude Threshold	0.90649666
10	0.90649666
10.25	0.90912771
10.5	0.91287189
10.75	0.91600891
11	0.91833637
11.25	0.91853876
11.5	0.92177697
11.75	0.92278891
12	0.92380085
12.25	0.92390204
12.5	0.92359846
12.75	0.92400324
13	0.9246104
13.25	0.92440801
13.5	0.92359846
13.75	0.92349727
14	0.92339607
14.25	0.92349727
14.5	0.92309249
14.75	0.92197936
15	0.92086622
15.25	0.91995547
15.5	0.91833637
15.75	0.91722323
16	0.91590771
16.25	0.91368144
16.5	0.91165756
16.75	0.90943129
17	0.90791338
17.25	0.90558591
17.5	0.90487756
17.75	0.9023477
18	0.90002024

Figure 5.3: Accuracy at different magnitude thresholds. Red values indicate high accuracy, while green values indicate lower accuracy. The highest accuracy of approximately 92.5% was with a threshold of 13.0 μT.

5.3 Machine learning and detection algorithms

We used scikit-learn [22] machine learning classifiers to improve upon the accuracy of the magnitude-based threshold algorithm discussed in Section 5.2. We sought to use simple machine learning models to improve the prediction scores for the following reasons:

- Using simple algorithms allows us to put the parking space occupancy decision logic on a computationally constrained sensor
- Sensors can then relay "vacant" or "occupied" to a central server, circumventing the necessity for the central server to constantly make occupancy decisions for each space
- We are only working with four features (X, Y, Z, and spot number), and more complex models like deep learning neural networks might "learn" noise.

We recognize that there are trade offs to making the occupancy decision on the remote device vs. on a server. The advantage to server-side processing is that more data can be compiled, more complex modeling algorithms are possible, and re-training the whole system can be done on the server. The disadvantages to this approach are that there exists a potential for the server to get flooded with input data; processing city-wide parking data would require significant computational power and storage. Putting the occupancy decision on the device simplifies the server-side functions; the server is just a database that stores the available and current parking availability of those spots.

Moving back to our work, to train and test these models, we used the processed data (with baselines subtracted). For our features, we input the X, Y, Z, and spot number values for each reading; our target was just vacant or occupied which we set to a binary value (0 or 1 respectively). For the training data, we used 70% of the total corpus, and for the testing data we used 30% of the total corpus. Finally, we ran each model with a 10-fold cross validation (i.e. 10 different subsets of training and testing data).

We then tested three different classification models:

1. Gaussian Naive Bayes
2. Decision tree
3. Random forest.

5.3.1 Detection algorithm: Gaussian Bayes (naive)

We trained a naive Gaussian Bayes (GB) detection algorithm [23]. This algorithm assumes a strong naive independence between features. The model also assumes a Gaussian distribution of continuous data and classifies by minimizing the sum of squared errors.

We observe Gaussian Bayes 10-fold cross validation results consistent at roughly 95% accuracy. For assessing each of these models, we examined the confusion matrix (which plots the true positives, false positives, true negatives, and false negatives). We also examine the precision, recall, f-1 score, and support for each of these models. We show the confusion matrix for the Gaussian Bayes model in Figure 5.4.

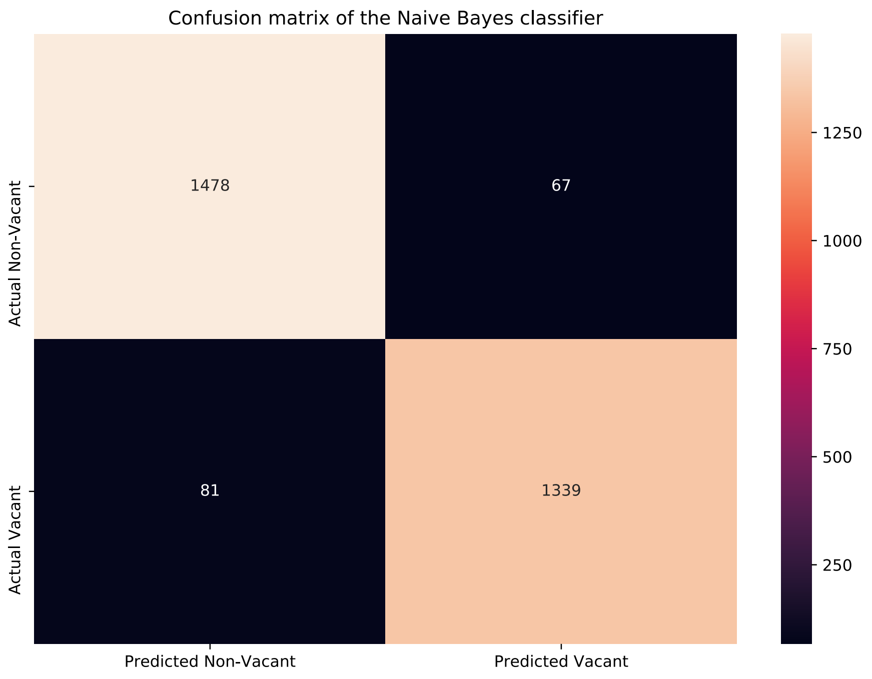


Figure 5.4: Confusion matrix plotting TP, FP, TN, FN for the GB model.

The classification performance is shown in Figure 5.5.

	Precision	Recall	F-1 Score	Support
Vacant	0.95	0.96	0.95	1545
Occupied	0.95	0.94	0.95	1420

Figure 5.5: Precision, recall, f-1 score, and support for GB model.

Naive Bayes Classifier accuracy results of 10-fold validation [0.95075885, 0.9477234, 0.94772344, 0.94637437, 0.95042159, 0.95514334, 0.94198988, 0.95042159, 0.94940978, 0.94974705]. **Average accuracy score 0.9489713322091063**.

This first model leveraging a simple machine learning algorithm could easily be run on a computationally constrained sensor and improves on the simple magnitude threshold algorithm by 2.43%.

5.3.2 Detection algorithm: decision trees

The next classification model we implemented is a decision tree. This model, as its name suggests, takes the form of a tree with weights/consequences for each branch. In our modeling, we maintained caution with the model potentially fitting to "noise" in the data. Therefore we tuned the parameters through several iterations to provide a more rigorous approach to this particular model. The parameters we ultimately tuned include:

- max_depth
- min_samples_split
- max_leaf_nodes.

Although, our first model of a decision tree we left with the default scikit-learn parameters, thus allowing the tree to make the highest fit possible to the data. Below we present the outcomes in Figures 5.6 and 5.7

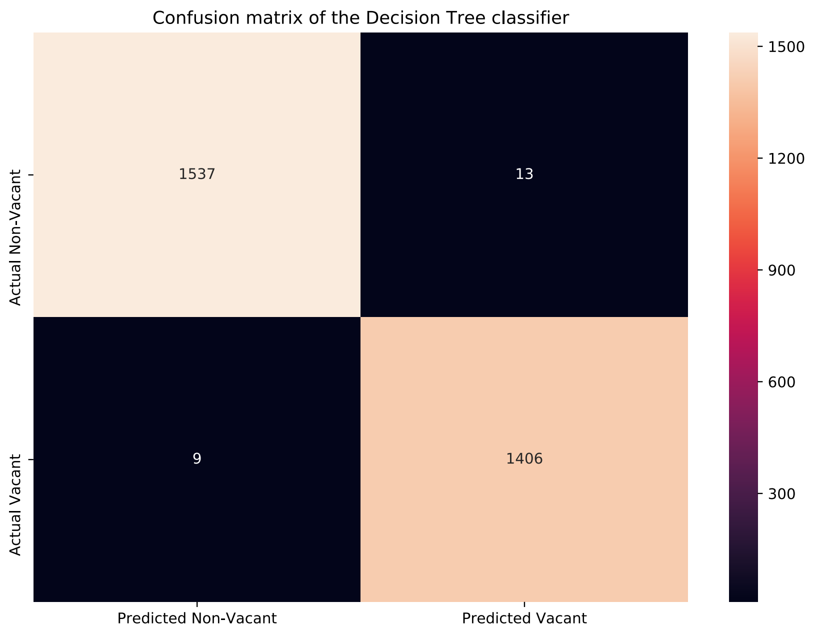
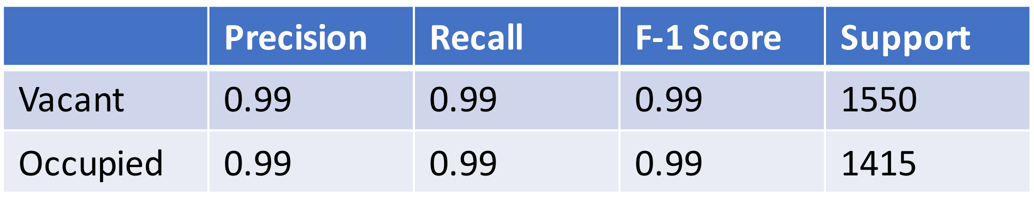


Figure 5.6: Confusion matrix plotting TP, FP, TN, FN for the decision tree optimal fit model.



	Precision	Recall	F-1 Score	Support
Vacant	0.99	0.99	0.99	1550
Occupied	0.99	0.99	0.99	1415

Figure 5.7: Precision, recall, f-1 score, and support for decision tree model with default parameters.

Decision tree classifier accuracy results of 10-fold validation [0.99224283, 0.98718381, 0.99359191, 0.98954469, 0.99123103, 0.99021922, 0.99460371, 0.99359191, 0.99392917, 0.99325464]. **Average accuracy score 0.9919392917369307**.

From the modeling results, we observe that this fit yields a very high accuracy of 99.2%, with only 9 false positives and 13 false negatives. While this fit is certainly highly precise and accurate, we recognize that there is the potential for the model to fit to the noise; it also yields a huge tree that a computationally constrained device might have trouble running. Therefore we analyze the resulting tree with the default model parameters shown in Figure 5.8.

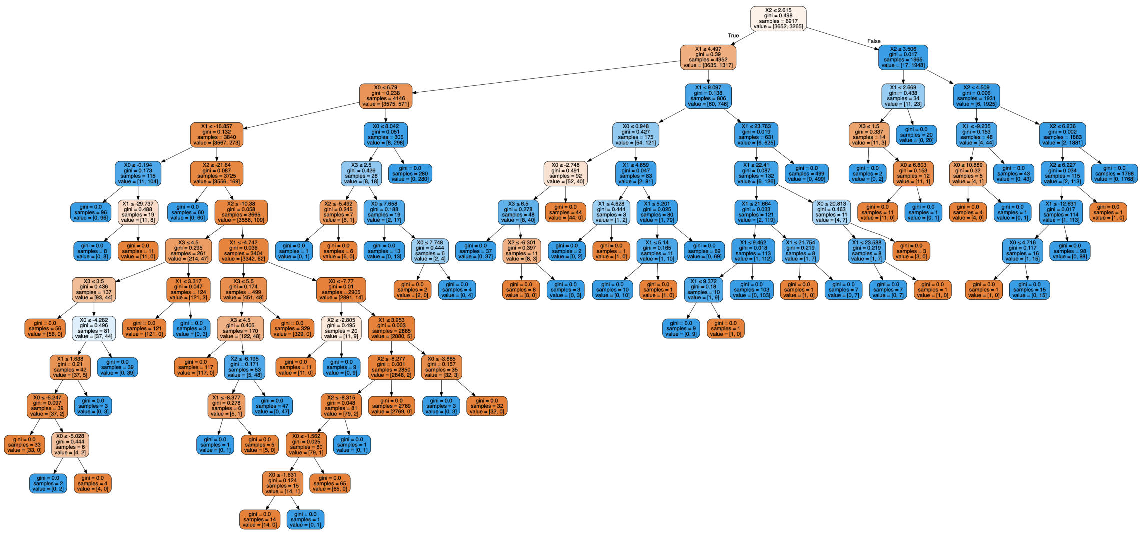


Figure 5.8: Tree graph for the "default parameters" decision tree shows high complexity and deep maximum depth.

We observe that the decision tree with maximum depth and fit, predicts the test data with 99% accuracy, and because we use 10-fold cross validation we gather this accuracy from the average of 10 different subsets of training and testing data. We also adjusted the test_size (i.e. how much of our data we save for testing) to be 0.6 of the total corpus, and we still observe 99% accuracy. This suggests that the model is able to accurately determine the true vehicle occupancy of a space. Our concern, however, is that the model may not generalize well enough (the cross-fold validation suggests otherwise), and the complex tree may not run well on a computationally constrained sensor.

We then adjust the parameters to tune our decision tree model, with the goal of simplifying the tree complexity. Below we present the Figure 5.9 and 5.10 for a decision tree with with max_depth = 8, min_samples_split = 50, and max_leaf_nodes=10.

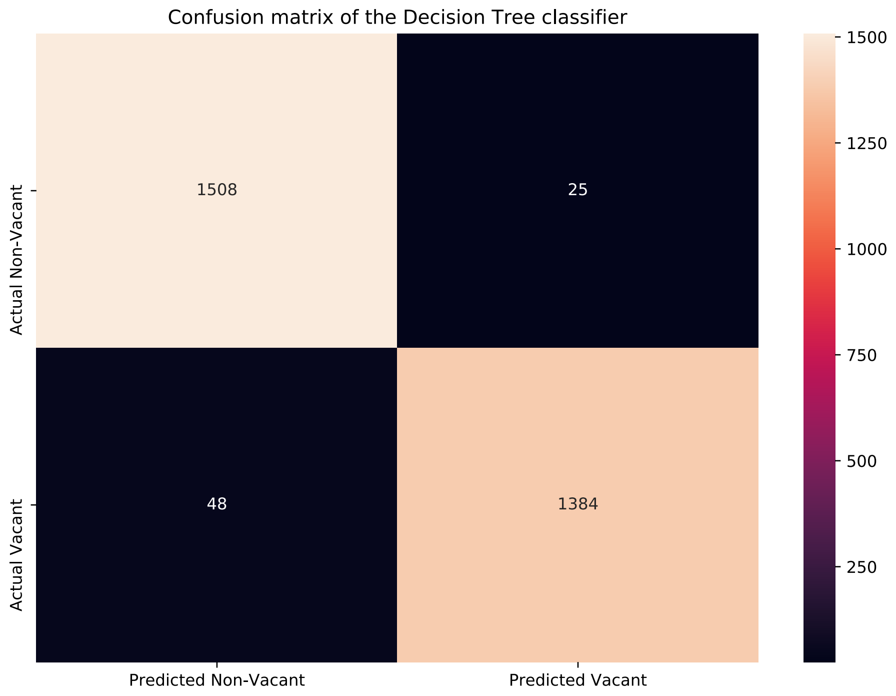
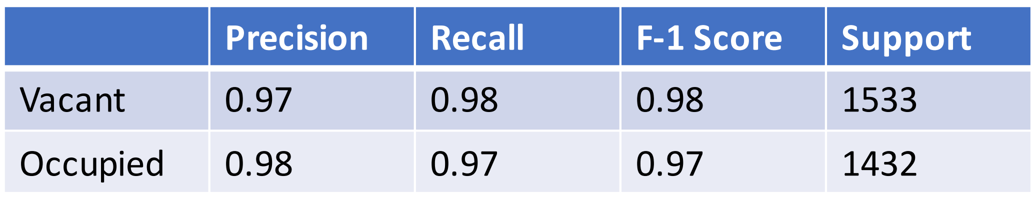


Figure 5.9: Confusion matrix plotting TP, FP, TN, FN for the decision tree tuned fit model.



	Precision	Recall	F-1 Score	Support
Vacant	0.97	0.98	0.98	1533
Occupied	0.98	0.97	0.97	1432

Figure 5.10: Precision, recall, F-1, and support for decision tree tuned fit model still shows better performance than the naive GB algorithm and the magnitude-method for detecting vehicles.

Decision tree classifier accuracy results of 10-fold validation [0.97133221, 0.97065767, 0.97200675, 0.9682968, 0.96492411, 0.97369309, 0.96526138, 0.9682968, 0.97234401, 0.96593592]. **Average accuracy score 0.969274873524452**.

The decision tree model with adjusted parameters still maintains a high accuracy of prediction at 96.9%. We also present a much simpler tree output in Figure 5.11.

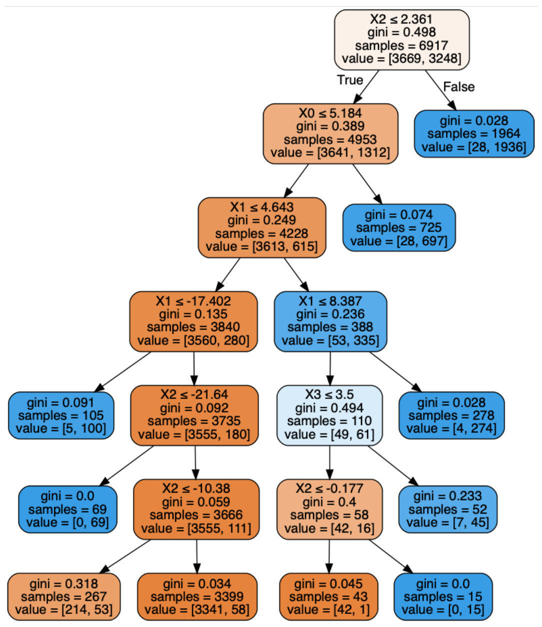


Figure 5.11: Tree graph for the tuned fit decision tree shows lower complexity and smaller max_depth.

With tuned parameters we observe a much less complex tree that gives us more reassurance we could both run this particular model on a computationally constrained device, and we still maintain high accuracy at approximately 97%. This score improves upon both the magnitude-based algorithm and the Gaussian Naive Bayes algorithm.

5.3.3 Detection algorithm: Random Forest

The last classification model we implemented is a Random Forest classification model which builds multiple decision trees then aggregates predictions (i.e. similar to a panel of judges as opposed to one judge). As in the simpler decision tree model in Section 5.3.2 above, we adjusted maximum_depth=8, number_of_estimators=100 (trees), and min_samples_split=50 of the forest of less complex trees. This model presents 10-fold cross validation results consistent at roughly 98% accuracy. We present the Figures 5.12 and 5.13 below.

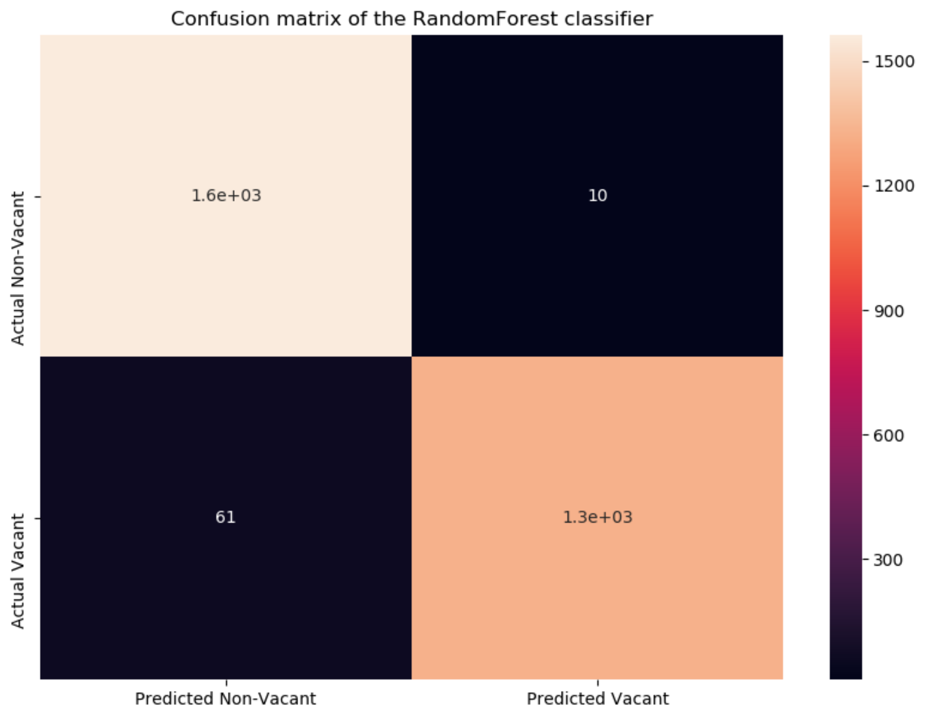
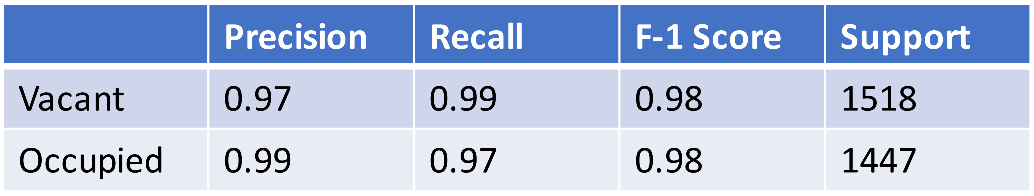


Figure 5.12: Confusion matrix plotting TP, FP, TN, FN for the Random Forest tuned fit model shows slightly better performance than one decision tree with the same parameters.



	Precision	Recall	F-1 Score	Support
Vacant	0.97	0.99	0.98	1518
Occupied	0.99	0.97	0.98	1447

Figure 5.13: Precision, recall, f-1 score, and support for Random Forest tuned fit model

Random Forest classifier accuracy results of 10-fold validation [0.97875211, 0.97841484, 0.97976391, 0.9767285, 0.98077572l, 0.98043845, 0.98010118, 0.98313659, 0.97942664, 0.97234401]. **Average accuracy score 0.9789881956155142**.

The Random Forest model data again shows that these particular classification models perform with a high degree of prediction accuracy. While the individual trees were limited in depth and number of trees was kept somewhat small, the more complex forest of trees only outperformed a single simple tree by a small amount (98% vs. 97%). Our key take way message is that the models are hitting diminishing returns – the extra complexity of many trees only results in a small increase in accuracy compared with a single simple tree.

5.3.4 Machine learning classification model conclusions

When examining all of the approaches, we see that the models using trees to classify (decision tree and Random Forest) perform at high accuracy. If the microcontroller has limited computational power, then we would recommend using the simple decision tree model, while if the controller is powerful, we would recommend the training and use of a Random Forest (or possibly a large single decision tree). Our models train for specific spots in specific lots; these are not generic models that someone can plug in anywhere (although the magnitude model is general and can be used anywhere). In the real world, we would still need to gather training data for each space. Gathering training data would involve placing a sensor in each space, watching each spot manually, recording data, and then running classification models. See Section 6.2 for more discussion surrounding these factors.

5.4 Optimal sensor placement

Recall our objective to determine the optimal sensor placement within a parking space. We thus sought a means of evaluating this placement. To do so, we ran a decision tree model for each individual sensor and plotted a heat map of accuracy as shown in Figure 5.14.

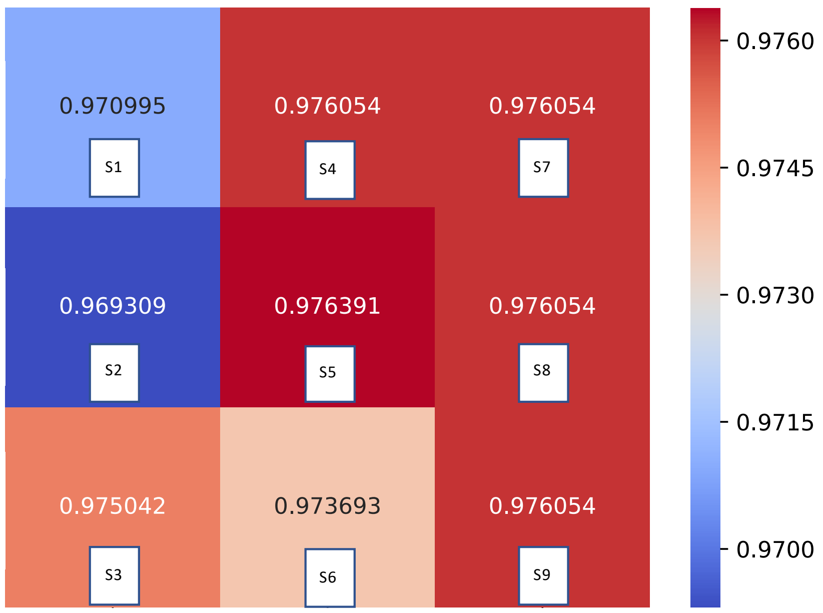


Figure 5.14: Heat map of prediction accuracy by sensor location within a parking space. All locations performed nearly identically.

We observe that there is a marginal improvement in accuracy by placing the sensor either directly in the center of the parking space, or slightly offset to the right. The improvements are roughly 0.6%. Furthermore, we note that these differences vary; if we run this model again, we could observe a slightly different heat map. Accordingly, our conclusion is that so long as the sensor is placed centrally in the parking space we will observe an accuracy of roughly 97%. We consider that perhaps the surface area of the sensor board was not large enough to capture differences in sensor location. Another factor is the actual variance of vehicles' parking; every time we placed the sensor board in its designated spot the vehicle above it could be anywhere between the white lines. In other words, the cars were not perfectly parked in the exact same positioning in the parking space for every data reading. We used real parking data from real people and as such there is the potential that this variance in parking limited the degree to which sensor placement is important, which is why we observe that each sensor predicts with about the same accuracy. Finally, we conclude that central placement of the sensor in the parking space is essential, but past that more data collection and controlled parking scenarios are required.

Chapter 6

Future Work, Limitations, and Conclusions

6.1 Future work

In future work using these magnetometer-based vehicle sensors, we would undertake an assortment of tasks. One of which would be to collect data from more parking arrangements (e.g. parallel, street parking, etc.). In this stage, we would ideally physically install sensors across campus and aggregate more data. We would also architect a full system for end-to-end functionality, which would necessitate both evaluating power constraints and prototyping a "long-term" sensor. In this implementation we would consider different technology to relay data (e.g., Bluetooth, radio, WiFi, LoRa). Finally, the system would aggregate sensor data on a central server for applications to integrate with navigation apps.

6.2 Limitations

The primary limitation in our work is that we only analyzed one surface lot. If we were to have more time, we would collect across every possible parking arrangement (parallel parking, garage parking, street parking, etc.). Furthermore, testing across multiple lots and locations would further show the efficacy of such a solution.

We also collected our data using the mobile sensor board. There is the chance we introduced error if the board was not placed in the exact same space. Physically installing these sensors (so they could not move at all) would reduce the chance of such error.

Another limitation is that we currently model using the space number as a feature. If we were able to exclude this from our modeling, we may be able to generalize the detection algorithms to all surface lots. Developing a universal algorithm would reduce the overhead required for the installation and training of a system. The only initial data to collect would be a baseline reading for a parking space.

We would seek to prove the viability of this technology in an end-to-end system as mentioned in Section 6.1. If we could succeed in implementing such a system that would provide an even better foundation to go out and actually implement these sensors in the real-world for smart-city applications like using parking availability to integrate with a navigation map service (i.e. Google maps).

Finally, a disadvantage of magnetometer-sensors is the requisite for close proximity to the target. As distance from the sensor increases, the magnetic dipole decreases with an inverse cube [24]. Therefore, proximity to the vehicle is key for accurate detection.

6.3 Conclusions

The algorithms and sensor placements that we tested in a surface lot predict predict the data with a high degree of accuracy. Table 6.1 below shows a summarized performance for each of these models.

Model	Avg. accuracy
Magnitude	92.5%
Gaussian NB	94.9%
Decision Tree (simple)	96.9%
Random Forest	97.9%
Decision Tree (default parameters)	99.2%

Table 6.1: Accuracy of each detection model. All models performed well into the 90th percentiles. More complex models show diminishing accuracy improvements over simpler models.

We observe that the decision tree using the default parameters performs the best of all models. Figure 5.7, however, shows this tree is highly complex. All models, however, perform well with accuracies well into the 90th percentile. This result suggests these low cost magnetometer-based sensors are effective for detecting vehicles. We also note that with simple detection algorithms we can put the decision logic of whether the parking space is occupied on the sensor, whereas complex models such as deep neural networks may quickly surpass the computing capabilities of a small microcontroller. We show through our testing that magnetometer-based technology is a cost effective solution for vehicle detection. With magnetometer sensors available for $1.09 [25] when purchased wholesale (quantity of 2,500), the overall cost is not significant when compared to some of the other technologies like detection cameras. Magnetometer-based technology offers advantages over several other of the fixed sensors. Primarily, a magnetometer is not sensitive to ambient changes in the environment such as temperature, rain, or snow. Since magnetometers are not sensitive to such environmental factors they offer advantages over other fixed sensor technologies for the following reasons:

- Cameras infrared sensors, microwave Sensors, ultrasonic sensors, and light sensors all face challenges with environmental variables and occlusion
- Magnetometers only alert to changes in the magnetic field from metal ob-

jects, while other sensors can falsely alert to non-vehicle signals

- Magnetometers are cheaper than other solutions like camera recognition.

We note that different levels of accuracy are required for certain implementations. For simple availability of parking spaces on city blocks used to direct motorists to blocks where parking is likely to be found, 90% accuracy is sufficient [13]. For example, if there are anywhere from 10-20 spaces on a city block, then if a system can correctly predict the status of 9/10 or 18/20 of those spaces, it would suffice for directing drivers to open parking.

Another application is the use of such detection technology for ticketing purposes. A problem facing parking meters worldwide is that a user can simply “feed” the meter (e.g., add more money) allowing them to stay in a single spot far past the legal limit. Were this technology used to determine if a violator has stayed past their permitted allotment of time, then 99.99%+ accuracy is a requisite, else the parking enforcement officers would quickly be issuing a large number of potentially invalid tickets.

We also set out to determine the best placement for a sensor in a parking space. As discussed in Section 5.4, placing the sensor centrally in the parking space is key, beyond that, marginal improvements are observed by placing the sensor more specifically under the vehicle. We note that a limitation of our data collection is that we did not control for how a vehicle parks in a space (lots of variability for a driver parking in a space). If we had more time and resources, perhaps we could control for every data collection, but we did capture real-world parking data and succeed in showing that this technology works well.

6.4 Closing remarks

Here I close my culminating experience. The past six or so months have been incredibly reformative, and the work on this project has been both an incredible challenge as well as an extremely rewarding experience. I've learned more than I ever imagined I would about parking. I also used nearly every aspect of my computer science knowledge, and I learned a tremendous amount of other skills along the way. We proved that these low cost sensors work, and it doesn't matter exactly where you place them. My hope is that all this work will serve to help make our cities smarter and solve some aspect of the great conundrum surrounding parking sensors. Finally, another thank you to Tim Pierson for all of your time and energy helping make this project into such a successful undertaking. So long, Dartmouth.

Luke Hudspeth

References

[1] D. C. Shoup, *Planning for Free Parking: The High Cost of Free Parking*. American Planning Association, 2011. 2

[2] Texas Transportation Institute, *Urban Mobility Report*. Texas A&M University, 2007. 2

[3] SFPark, "On-street Parking Census Data and Map," April 2014. `http://sfpark.org/` [Accessed Nov 26, 2018]. 2, 3, 11

[4] LA Express Park, "LA Express Park." `http://www.laexpresspark.org/` [Accessed Nov 26, 2018]. 3

[5] NWAVE, "Smart parking solutions overview," June 2018. `https://www.nwave.io/news/plug-and-play-smart-parking/` [Accessed Nov 26, 2018]. 3

[6] CIO Techie – Enterprise Global Business Technology Magazine, "VIMOC lands the smart parking contract in North County San Diego." `https://ciotechie.com/news/vimoc-lands-the-smart-parking-contract-in-north-county-san-diego/` [Accessed May 27, 2019]. 6

[7] G. Revathi and V. S. Dhulipala, "Smart parking systems and sensors: A

survey," in *International Conference on Computing, Communication and Applications (ICCCNT)*, pp. 1–5, IEEE, 2012. 6, 7, 8, 9, 10, 11

[8] S. A. Shaheen, C. J. Rodier and A. M. Akin, "Smart parking management field test: A Bay Area Rapid Transit (BART) district parking demonstration," 2005. 7

[9] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," in *International Symposium on Collaborative Technologies and Systems (CTS)*, pp. 48–57, IEEE, 2008. 7, 9, 10, 11

[10] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet: drive-by sensing of road-side parking statistics," in *Proceedings of the International conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 123–136, ACM, 2010. 7, 8

[11] Y. Shen, G. Chen, X. Xu, C. Xu, G. Shen, and J. Li, "Poster: A VLC solution for smart parking," in *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pp. 579–581, ACM, 2017. 10

[12] InvenSense, "MPU9250." `https://robu.in/product/mpu9250-9-axis-attitude-gyro-accelerator-magnetometer-sensor-module/` [Accessed: May 20, 2019]. 10, 15, 18

[13] San Francisco Municipal Transportation Agency, "SFPark pilot evaluation." `http://sfpark.org/` [Accessed May 23, 2019]. 11, 59

[14] ST Electronics, "LSM303." `https://cdn-shop.adafruit.com/datasheets/LSM303DLHC.PDF` [Accessed May 18, 2019]. 13, 14, 15, 17

[15] Honeywell, “HMC5883.” https://media.digikey.com/pdf/Data%20Sheets/Honeywell%20PDFs/HMC5883L.pdf [Accessed May 18, 2019]. 13, 14, 18

[16] Arduino, “Arduino Uno.” https://store.arduino.cc/usa/arduino-uno-rev3 [Accessed Dec 15, 2018]. 15, 16

[17] Node MCU, “ESP 8266.” https://nodemcu.readthedocs.io/en/master/nodemcu-documentation [Accessed Jan 4, 2019]. 16

[18] Nordic Semiconductor, “nRF2401.” https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf [Accessed Dec 15, 2018]. 25

[19] Arduino, “Arduino Nano.” https://store.arduino.cc/usa/arduino-nano [Accessed Jan 4, 2019]. 25

[20] JLCPCB, “PCB prototype and PCB fabrication manufacturer.” https://jlcpcb.com [Accessed May 26, 2019]. 26

[21] Fritzing, “Fritzing, electronics made easy.” http://fritzing.org/home/ [Accessed Dec 15, 2018]. 26

[22] scikit-learn, “scikit-learn.” https://scikit-learn.org/stable/ [Accessed May, 2019]. 43

[23] R. Ng, “Gaussian Naive Bayes.” https://www.ritchieng.com/machine-learning-gaussian-naive-bayes/ [Accessed May 23, 2019]. 45

[24] Millersville University Physics Department, “Magnetic fields varying as an inverse cube.” https://www.millersville.edu/physics/experiments/023/index.php [Accessed May 29, 2019]. 57

[25] Mouser Electronics, "IIS2MDCTR pricing." https://www.mouser.com/ProductDetail/STMicroelectronics/IIS2MDCTR?qs=sGAEpiMZZMve4/bfQkoj+NsA/YLrKsWenI+NJIvEAfE=.STMicroelectronics [Accessed May 29, 2019]. 58

[26] Arduino, "Arduino IDE." https://www.arduino.cc/en/Main/Software [Accessed Dec 15, 2018].

[27] JetBrains, "PHP Storm." https://www.jetbrains.com/phpstorm/ [Accessed Dec 15, 2018].

[28] JetBrains, "PyCharm." https://www.jetbrains.com/pycharm/ [Accessed Dec 15, 2018].

[29] J. Woetzel, J. Remes, B. Boland, K. Lv, S. Sinha, and G. Strube, "Smart cities: Digital solutions for a more livable future," June 2018. https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/smart-cities-digital-solutions-for-a-more-livable-future [Accessed May 22, 2019].