Miner-Town: Self-Driving Robotics Testbed For Vehicle-To-Grid Simulation

Carlos Adolfo Cortes Pliego

University of Texas at El Paso

Follow this and additional works at: https://scholarworks.utep.edu/open_etd

Part of the Computer Engineering Commons

Recommended Citation
https://scholarworks.utep.edu/open_etd/3596

This is brought to you for free and open access by ScholarWorks@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.
MINER-TOWN: SELF-DRIVING ROBOTICS TESTBED FOR
VEHICLE-TO-GRID SIMULATION

CARLOS ADOLFO CORTES PLIEGO
Master’s Program in Computer Engineering

APPROVED:

________________________________________
Robert C. Roberts, Ph.D., Chair

________________________________________
Patricia Nava, Ph.D.

________________________________________
Virgilio Gonzalez, Ph.D.

________________________________________
Ruimin Ke, Ph.D.

________________________________________
Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School
Copyright ©

by
Carlos Adolfo Cortes Pliego
2022
Dedication

It is hard to include and say “thank you” when there are so many people that were part of my journey through my college and graduate experience. Obviously, this thesis is a thank you to my parents and family, who were powerful role models, taught me love and kindness, shaped the values that made me who I am and were a constant support in times of need.

The person most responsible for this thesis becoming a reality is my girlfriend Alexa – my best friend, and partner in life. She makes my life complete, and without her anything of this effort could have been possible.
MINER-TOWN: SELF-DRIVING ROBOTICS TESTBED FOR VEHICLE-TO-GRID SIMULATION

by

CARLOS ADOLFO CORTES PLIEGO, B.S.EE

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at El Paso
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO
August 2022
Acknowledgements

I would like to acknowledge Dr. Robert C. Roberts for guiding me throughout my research and thesis. He was always available whenever I experienced any roadblock and answered every question I encountered. Dr. Roberts was a true mentor and was able to lead me to a successful and enriching research experience. If it were not for him, I would have never done any robotics investigation, which turned out to be thought-provoking and fascinating.

I would also like to give special recognition to Dr. Patricia Nava. Thanks to her, I started my research journey in artificial neural networks architectures and thanks to her passion and commitment I decided to pursue my master’s in computer engineering.

Finally, I would like to acknowledge the UTEP College of Engineering for their continuous support throughout my bachelors and master’s degree.
Abstract

Autonomous vehicles and Vehicle-to-Grid (V2G) technology bring promising implications in boosting energy efficiency, helping the environment, improving our productivity, and have the potential to stabilize the grid during peak times and reduce car accidents. However, implementing and testing these complex novel technologies in the real world comes with high risks and investment. For these reasons, there is the need to research, test, and validate these theories in a compact and controlled environment at minimal cost. This thesis presents a modular autonomous vehicle testbed for the exploration of Vehicle-to-Grid and charging activities in pedestrian filled environments such as a University campus. Previous research on low-cost robotic platforms has reported promising results on control systems, navigation, and object detection; however, the implementation of pedestrian recognition and autonomous docking for V2G chargers remains to be accomplished. By extending the capabilities of existing environments and platforms including Duckietown and the Nvidia Jetbot robot; road following, pedestrian recognition, and autonomous charger docking features are added to enable a Vehicle-to-Grid testbed. Neural networks, supervised learning (classification and regression), and transfer learning techniques are utilized to integrate these functionalities to the testbed. Additionally, the use of fiducial markers is investigated to get a better perception and depth for vehicle to object alignment for camera-based sensing. Through the realized testbed, it is hoped that researchers can understand and overcome the obstacles that these technologies bring with it, to cost effectively learn, develop, and teach the necessary solutions to advance both autonomous vehicles and V2G technology, not only in the classroom, but also to industry.
# Table of Contents

Dedication ........................................................................................................................................ iii

Acknowledgements ......................................................................................................................... v

Abstract ........................................................................................................................................ vi

Table of Contents ............................................................................................................................. vii

List of Tables ...................................................................................................................................... xi

List of Figures .................................................................................................................................... xii

Chapter 1: Introduction ...................................................................................................................... 1

  1.2 Organization ............................................................................................................................... 3

Chapter 2: Vehicle-to-Grid (V2G) and motivation for testbed ......................................................... 5

  2.1 Vehicle to grid background ....................................................................................................... 5

    2.1.1 Defining V2G ....................................................................................................................... 5

    Advantages .................................................................................................................................. 6

    Barriers ....................................................................................................................................... 7

    2.1.2 How does V2G work? .......................................................................................................... 8

    Actors and roles of V2G ............................................................................................................... 8

    2.1.3 Most common fields of research around the V2G ............................................................. 10

    Integration of V2G with Renewable Energy Systems and Smart Grid ......................... 10

    Charging Stations ....................................................................................................................... 10

  2.2 Motivation .................................................................................................................................. 11

    2.2.1 Vehicle-to-Grid in early stages ......................................................................................... 11

    2.2.2 Moving towards a scalable modular testbed for V2G implementation .................... 11

    2.2.3 Implementing Autonomous Vehicles with V2G capabilities ...................................... 12

  2.3 Proposed Solution ...................................................................................................................... 12

    2.3.1 Scale up from testbed to on-campus implementation ....................................................... 13

  2.4 Research Questions .................................................................................................................... 13

  2.5 Chapter Summary and transition to Autonomous Vehicles .................................................. 14

Chapter 3: Autonomous Vehicles ..................................................................................................... 15

  3.1 How do Electrical Vehicles work? ............................................................................................ 15
3.2 Autonomous Vehicles Background ................................................................. 15
  3.2.1 Defining Autonomous Vehicles .............................................................. 15
      Advantages .................................................................................................. 16
      Barriers ..................................................................................................... 17

3.3 Sensors in Autonomous Vehicles ................................................................. 17
  3.3.1 Camera ................................................................................................... 18
  3.3.2 Radar ................................................................................................. 18
  3.3.3 Lidar .................................................................................................... 19

3.4 Chapter summary and the use of the camera sensor in low-cost platforms ........ 19

Chapter 4: State of the Art & Machine Learning Techniques .................................. 20
  4.1 Simulated Environments ............................................................................. 20
      4.1.1 Duckietown ...................................................................................... 20
      4.1.2 Jetbot AI Robot .............................................................................. 22

  4.2 Machine Learning Techniques ................................................................... 23
      4.2.1 Supervised Learning in Data Classification ....................................... 23
          How does supervised learning work? ............................................... 23
          How effective can this approach be for pedestrian classification? ........ 25

      4.2.2 Transfer Learning for Visual Categorization ................................. 25
          How transfer learning works? .......................................................... 26
          Why use Transfer Learning? .......................................................... 26

      4.2.3 Data Splitting .................................................................................. 27
          What is the best split ratio? .............................................................. 28

      4.2.4 SoftMax activation function ......................................................... 28
          Sigmoid ............................................................................................... 28
          ReLU ................................................................................................. 28
          SoftMax ......................................................................................... 29

      4.2.5 Cross-entropy loss function ......................................................... 30

      4.2.6 MSE loss function ......................................................................... 31

      4.2.7 Image-Coordinate Probabilistic Regression ................................. 31
          Applications of regression models in computer vision ...................... 31

      4.2.8 Automated robot docking ......................................................... 32
          Line Following Algorithm .............................................................. 32
4.3 Pre-trained models ........................................................................................................33
  4.3.1 Alexnet ......................................................................................................................34
  4.3.2 Resnet .......................................................................................................................34
  4.3.3 Pre-trained models’ comparison .................................................................................35
4.4 Chapter summary ..........................................................................................................35

Chapter 5: Research Design and Methodology ................................................................36
  5.1 Aims and Objectives ....................................................................................................36
  5.2 Approach .....................................................................................................................36
  5.3 Simulated Environment Design ................................................................................37
    5.3.1 UTEP Campus Inspiration ..................................................................................37
    5.3.2 Road Materials .....................................................................................................38
      Interlocking black tiles .................................................................................................39
      Waveshare Jetbot AI Kit ............................................................................................39
    5.3.3 Road Design ..........................................................................................................40
    5.3.4 Bill of Materials .....................................................................................................40
  5.4 Chapter Summary ........................................................................................................40

Chapter 6: Road Following Module ..................................................................................41
  6.1 Objective .....................................................................................................................41
  6.2 Approach and Framework .........................................................................................41
  6.3 Dataset Creation .........................................................................................................42
  6.4 Training and Verification Results ...............................................................................42
  6.5 Inference Results .......................................................................................................43
  6.6 Odometry Constrains ...............................................................................................44
  6.7 Chapter summary .......................................................................................................45

Chapter 7: Object/Pedestrian Recognition .....................................................................46
  7.1 Objective .....................................................................................................................46
  7.2 Classification categories ............................................................................................46
    7.2.1 Pedestrians ..........................................................................................................46
    7.2.2 Traffic Signs .........................................................................................................47
  7.3 Module Approach and Framework ............................................................................48
  7.4 Data Collection .........................................................................................................49
List of Tables

Table 4.1: Technical specifications for NVIDIA Jetson Nano Developer Kit ......................... 22
Table 4.2: Performance Comparison between pre-trained models. Modified from [78] ............ 35
Table 7.1: CE Loss and Accuracy from the Training and Verification process ....................... 52
Table 7.2: Testing accuracy of pedestrian recognition model. ........................................ 55
List of Figures

Figure 2.1 Schematic of a V2G system - Fig. modified from [15] .......................................................... 9
Figure 4.1: Road inspired in Duckietown .................................................................................................. 21
Figure 4.2: High level scheme of how supervised learning works. Fig modified from [54] .............. 24
Figure 4.3: Illustration of transfer learning technique. Figure modified by [59] ............................... 26
Figure 4.4: SoftMax function implementation ....................................................................................... 29
Figure 4.5: Cross-Entropy simple schematic ......................................................................................... 30
Figure 4.6: Data collection method using camera and joystick. Fig. modified from [76] ................. 33
Figure 5.1: Miner-town modules and tasks ............................................................................................. 37
Figure 5.2: Proposed circuit road for autonomous robot navigation around UTEP’s campus .... 38
Figure 5.3: Road tile types ................................................................................................................... 39
Figure 5.4: Miner-town road design ........................................................................................................ 40
Figure 6.1: Road following Framework .................................................................................................. 41
Figure 6.2: Road following data collection process .............................................................................. 42
Figure 6.3: Training and validation graph results for road following................................................... 43
Figure 6.4: Road Following inference results .......................................................................................... 43
Figure 6.5: Robot behavior with open-loop motors ................................................................................ 45
Figure 7.1: Figurines used to simulate pedestrians .............................................................................. 46
Figure 7.2: Pedestrian figurines with real people's faces ...................................................................... 47
Figure 7.3: 9cm – wooden traffic signs to simulate stop, parking, and speed limit indicators ......... 47
Figure 7.4: Object/Pedestrian classification module framework .......................................................... 48
Figure 7.5: Data collection process with 125 images per class .............................................................. 49
Figure 7.6: Training and Verification with 50 samples, and 30 epochs ....................................... 50
Figure 7.7: Training and Verification graph with 50 samples, and 100 epochs ............................... 51
Figure 7.8: Training and Verification graph with 100 samples, and 100 epochs ............................. 51
Figure 7.9: Training and verification graph with 150 samples, and 100 epochs ............................ 52
Figure 7.10: Pedestrian detection results graph ................................................................................. 53
Figure 7.11: Pedestrian detection results with backlit ......................................................................... 53
Figure 7.12: Pedestrian detection comparison with Lego figurines ..................................................... 54
Figure 7.13: Pedestrian detection comparison with real people’s faces ............................................ 54
Figure 7.14: Traffic signs result graphs ................................................................................................ 56
Figure 7.15: Multiclass vs single class recognition .............................................................................. 57
Figure 8.1: Autonomous Docking Framework .................................................................................... 58
Figure 8.2: Parking Design Layout ......................................................................................................... 59
Figure 8.3: Charging Station Design ..................................................................................................... 60
Figure 8.4: Energy transfer stages between Jetbot and charging station ............................................. 61
Figure 8.5: Parking recognition performance in different scenarios .................................................... 62
Figure 8.6: Tested code for entering the parking lot .......................................................................... 63
Figure 8.7: Line following data collection and training and validation results .................................. 64
Figure 8.8: Autonomous docking success rate in ideal conditions ..................................................... 65
Figure 8.9: Error in cm at unsuccessful docking ................................................................................... 65
Figure 8.10: Examples of AprilTag families ......................................................................................... 66
Figure 8.11 Apriltag detection and XY coordinates of center ............................................................... 67
Chapter 1: Introduction

It is not a secret that through the years, humanity has made significant efforts to improve the quality of life of people without affecting our surroundings and environment. One of these efforts comes from the emergence of Autonomous Electric Vehicles (A-EVs) in recent years. These initiatives not only promise to reduce the large carbon emissions produced by oil- or diesel-based automobiles and contribute to improving air quality in towns and cities, but it also opens new research gaps.

Autonomous Electric Vehicles are coming to be the next generation of mobile transfer. They have come with many advantages and promising implications in the areas of Artificial Intelligence (AI), and renewable energy. But what if this technology could also form part of the next energy revolution? Bi-directional energy transfer – also known as vehicle-to-grid (V2G) charging – is an upcoming technology in Electric Vehicles (EVs) that not only allows them to take power from the grid to charge their battery, but it also gives them the capability to supply power back to the grid. These upcoming technologies inspires us to explore the use of autonomous electric vehicles as a charging alternative to reduce grid load during peak times, as well as a consumer-side mobile power source. However, the implications of implementing autonomous driving in an urban setting carries several risks worth considering. Some of these roadblocks and concerns include major urban planning changes, privacy and safety at stake, high-cost investment, time consuming legislations, etc. For these reasons, there is the need to conduct exhaustive testing in a controlled and simulated environment. Ideally, a scaled testbed would be available that can have relevant characteristics while allowing smaller, safer, cheaper, and faster testing. This could be of great benefit to the UTEP-El Paso Electric partnership, which looks to enhance energy research and improve education in El Paso region.
The purpose of this research is to study and explore the implementation of V2G Autonomous Electric Vehicles in urban environments by developing a portable scaled-based platform that would allow easier testing, research, and development of this technology. In this thesis the efforts are focused on four key aspects that will serve as a starting point of further development. First, practical road designs for autonomous driving are studied. By taking inspiration with the UTEP campus, the proposed platform will serve as a blueprint and testbed for road modifications needed to implement autonomous driving vehicles at the university. Next, the investigation and implementation of practical techniques like supervised learning & transfer learning are conducted, to understand if they are the ideal techniques for object and pedestrian recognition. Similarly, regression techniques for road following implementation are analyzed and the performance of such method is measured. Finally, different approaches like visual fiducial systems and line following techniques are explored and applied for autonomous docking into charging/discharging stations. Over the course of this work, it is learned the promising implications and benefits that autonomous vehicles, and V2G technology will bring to the society. The hope is that this testbed could form the foundations to allow the further development of V2G and self-driving technology across pedestrian based environments.

This research represents a great opportunity for the University of Texas at El Paso, their students, and community to be research pioneers in the upcoming energy revolution. It will also help strengthen the V2G and autonomous navigation concepts and many of its theories. The creation of a test bench where many research hypotheses or simulations could be proved in a scaled physical setting will just push knowledge beyond what it is know now and with it, many other discoveries and findings will be possible. Without mentioning that it will create a basis for further research in Machine Learning, Autonomous Navigation and Renewable Energies.
1.2 Organization

The rest of this document is organized as follows:

Chapter 2 first explains the definition and implications of vehicle-to-grid technology. It starts by clarifying how V2G works and what are the factors/roles of everyone involved in the process. It then discusses the benefits and barriers of such technology and explains how our study can serve to tackle those barriers. After understanding the concept of V2G, the chapter goes into the motivation and purpose of this research. It starts by explaining that there is a need for energy efficiency in the grid and how the population can contribute to reducing the load of it in peak times. From there, it talks about the idea of using autonomous vehicles and vehicle-to-grid technology to address this concern and move toward a renewable and sustainable grid. It then discusses the roadblocks to implement this idea and how the proposed solution of creating a testbed for simulation can help us overcome the current obstacles. Finally, the chapter presents some research questions that we aim to respond to at the end of the study.

Chapter 3 goes over the concept of Autonomous Vehicles. It explains their functionality, advantages and deep into the most common sensors used by existing vehicles. Next, the chapter punctuates the need of a robotic platform to simulate autonomous vehicles behavior. It then ends setting the basic requirements for the scaled based robot.

Chapter 4 goes through a literature review about the most popular environments and techniques used so far for autonomous driving and docking. It first investigates Duckietown and Jetbot which will serve as the base inspiration for the proposed simulated environment. Finally, techniques and approaches for autonomous navigation and autonomous docking are described and analyzed. These will serve as a foundation for the implementation process.

Chapter 5 describes the details of our proposed experimental approach and the simulated environment design. It explains the materials used to simulate the road, and it also states the 3
modules in which the study will be divided. From there, we discuss the proposed road design, and why this design would be needed for autonomous navigation/docking.

Chapter 6 includes the procedures, techniques, and results for the Road Following module. It is first described the variables, framework and logic used. Then, a conducted analysis of the training, verification, and testing results. Finally, the challenges and implications of each section are discussed.

Chapter 7 oversees the implementation of object/pedestrian recognition. The materials used for simulation are presented. Then, the framework is defined, and the machine learning logic is presented. Finally, extensive testing is done to gather accuracy metrics in the classifier.

Chapter 8 includes procedures, techniques, and results for the Autonomous Docking module. First, we describe the framework and logic used. Then we conduct a comparison analysis between the 2 proposed approaches results as well as the mix of both. Finally, the challenges and implications of the module are discussed.

Appendix A contains the code and dataset used for the Object Recognition module. Appendix B contains the code and dataset used for the Road Following module. Appendix C contains the code and dataset used for the Autonomous docking module. Finally, appendix D covers the bill of materials used for the creation of our platform.
Chapter 2: Vehicle-to-Grid (V2G) and motivation for testbed

In this chapter, the concept of vehicle-to-grid is introduced and explained more in detail. By understanding its advantages and barriers, basic functionality, actors and roles, and its most common fields of research, the reader will gain a deep understanding of this technology, leading to further research and development. Additionally, the chapter explains the motivation behind the development of a modular testbed for autonomous vehicles with V2G capabilities. Finally, the testbed requirements are presented which represents the bases of further chapters.

2.1 Vehicle to Grid background

2.1.1 Defining V2G

The idea of Vehicle-to-Grid, which was formally introduced for the first time by Kempton and Letendre [1], instructs the use of an EV’s battery to provide storage for the grid. This basic concept known as V2G technology relies upon the ability to do energy transfer between the energy stored in the EVs and the grid. The purpose of this came first with the idea of helping supply energy at times of peak demand. It is considered a meaningful change in EV (Electric Vehicles) space because this two-way transfer not only encourages a more active way of consuming energy, but it also sets the ground for potential economic ventures for drivers, grid operators, and EV manufacturers.

This proposed technology works under the well-known fact of the low utilization rate of private vehicles during a complete day. According with [2], vehicles are typically parked 90% of the time. A common day for many civilians consists of commuting to and from work or going for food or groceries. But most of the time, this big mobile battery is an asset that is just sitting there. Most people are only using a fraction of their battery capacity and that means there is a huge amount of spare stored energy that could help solve energy efficiency problems. The
many advantages that come from this technology sets V2G worth to consider. Next, it is presented potential advantages and barriers of this technology in different sectors of interest.

Advantages

The transportation sector has one of the biggest sources of air pollution and GHGs emission and every day million tone of CO2 and other harmful gases enter the atmosphere [3]. For that reason, migrating from the traditional fleet to EVs and consequently implementing the V2G system, is an influential step toward mitigating the negative effects of GHGs on earth. In [4], the author indicates that EVs can improve the self-sustainability of the buildings and reduce the amount of energy supplied by the grid over one year up to 40% in the scenarios considered.

Technically speaking, the first benefit that comes with V2G is through the improvement of energy storage for the power grid. Under these terms, V2G is low cost, has a high-power capacity, and has a fast response time with correspondingly high efficiency [5]. In addition, it is projected that chargers will improve their efficiency making V2G even more feasible [2]. Secondly, if the storage-related barrier of batteries is solved, then V2G can offer the grid a bunch of many important advantages. Some of them include improved power quality, voltage support, transmission congestion relief, energy demand shifting, increased electricity reliability, and wind and solar integration [6].

Finally, there are also many economic benefits for all the participants around the V2G process. According to [7], V2G can represent an economic solution where everyone benefits all the actors implicated in the V2G process. However, the benefits in this area will vary from one primary actor to the other. For EV owners, V2G can represent a venue for passive income. And for grid operators and society, V2G can provide a cheaper alternative to current market participants. Besides all this, V2G can also increase the cost-effectiveness of utility services, which at the same time will reduce expenses for grid operations and society. In [8], the author shows that although V2G can reduce the lifetime of EVs, it is more economical for vehicle owners and the grid operator.
**Barriers**

There are two primary technical challenges to the implementation of V2G: battery degradation, and charger efficiency. Battery degradation is one of the most popular concerns around V2G. In [7] it is mentioned that battery degradation can cause loss of capacity over time, which impacts an EV’s range capability. The battery degradation due to V2G is primarily because of cycling aging. Even though degradation may be minor (or even improve battery health), consumers may resist participating in V2G out of fear that the impacts will be worse than expected, especially given the high valuation of battery life and range in general.

Another drawback of V2G in electric cars is the placement planning charging stations. Even though you can charge an electric vehicle at home, finding a charging station can be a challenge as today’s infrastructure is not developed around this technology. Charging stations must be placed in specific and not always convenient locations. In [9], mention that perfect arrangement of EV charging stations is turning into an issue worth to consider. This problematic will continue as demand for charging stations increases. Furthermore, the authors in [10] addresses that the lack of infrastructure for charging EV’s poses new challenges to the distribution network infrastructure and distribution network operators.

Lastly, one of the biggest obstacles and risks of implementing this system is the highly required investment. In a cost-benefit study conducted by [11], the authors found that the infrastructure cost of V2G is significantly more than smart charging infrastructure. The EVs must be equipped with plug-in connectors and meeting equipment to measure the input and output power and send the level of battery charging to the operators. Totally, all these factors need an important level of investment and time to implement the V2G system efficiently.
2.1.2 How does V2G work?

Now that the concept, advantages, and barriers of V2G are known, it is pertinent to mention the practical process of V2G functionality. The process of vehicle-to-grid first starts the other way around. The car takes the energy from the grid to charge its battery pack. Once the battery is full of charge, there is demand, and the owner is willing to do so, a special inverter then starts the conversion of energy from the battery to the grid [12]. It is true that the energy from one EV is almost negligible to the grid. Nevertheless, if potentially millions of these batteries could be deployed at the same time, it could represent the major power generation advancement in the decade [13].

_Actors and roles of V2G_

According to [14], there are three different primary actors in the V2G system, each representing a part of the “V2G” abbreviation. The first most logical actor in the V2G system is the EV owner. Without the EV owner, any other role would matter because they are the ones that provide the "V" - the vehicle. Besides this vital role, the owner is then relatively passive in the V2G process. They would only need to be in communication with the aggregator to notify them when they will commute to their desired places, and at the end collect the money generated from the service given by their vehicle.

Next, an intermediary between the owner and the grid operators is needed, which are called aggregators. Their role is to balance the power sent between the vehicle and the grid. Besides, they are the primary point of communication for both the owners and the grid operators. For instance, they represent the "2" in V2G. Normally, this aggregator is operated by a third party, and they are the ones who decide which electricity markets can participate in and when to do so.
Finally, the third primary actor is the electricity grid operator, which represents the "G" in V2G. According to [7], the electricity system can be segregated into two actors. First, the national electricity grid operator, also known as a transmission system operator (TSO), and the local utility, also known as a distribution system operator (DSO). The role of the TSO is to transmit large-scale electricity to areas with high demand through high voltage lines. On the other hand, the DSO's role is to receive the electricity from the TSO and then distribute it to end-users.

This process can be appreciated by referring to Figure 2.1. The modified Figure from [15] shows the proposed power line and wireless control connections between vehicles and the power grid. The control signal from the grid operator (ISO) request for power to multiple vehicles. This signal could be sent directly to individual vehicles (top right), the office of a fleet operator (bottom right), or to a third-party aggregator which in turn would become the main distributor to others. These actors are crucial for any interested to conduct research around vehicle-to-grid. Next, common fields of research are reviewed to perceive potential characteristics to add into a testbed.
2.1.3 Most common fields of research around the V2G

Integration of V2G with Renewable Energy Systems and Smart Grid

One of the reviewed solutions in [16] for solving environmental problems was migrating from using petroleum fuels and power plants to establishing the RESs (Renewable Energy Systems) for generating electric power. Especially photo-voltaic and wind power can be used for generating electric power without any air pollution and avoid GHGs emissions. But one of the biggest weaknesses of this system is that wind and solar energy are unpredictable and intermittent in nature. For that reason, the author reviews the integration of V2G and renewable energies as a solution to the problem. Similarly, in [17], the author proposes a practical solution to deal with challenges of integrating renewable energy sources and electric vehicles into the electric grid. A testbed designed for V2G implementation can bring significant value to this area of research. Renewable energy systems are completely scalable, and with a controlled environment the integration with V2G could be easily tested.

Charging Stations

As it was mentioned before, the study of charging stations has been one of the primary focal points around the research community. As mentioned in [18], the focus is to find the optimal allocation of charging stations based on economic benefits and grid impacts. Moreover, the proposed platform for autonomous navigation could be adapted to these V2G charging stations. Then, many tests around optimal allocation could be cheaper and easier to identify. Moreover, efforts to scale and implement tactless-based smart charging stations into the testbed can be done. Referring to [19], these types of charging stations are fissile and scalable for V2G implementation.
2.2 MOTIVATION

2.2.1 Vehicle-to-Grid in early stages

Even though V2G technology shows promising implications for redefining the grid as it is known, there are still many factors that prevent these theories from being fully implemented in a practical setting. First, the infrastructure cost could be too high for the implementation of this technology. Making major urban planning changes possible for experimentation could be costly, dangerous, and even counterproductive if it is not well executed. According with [20], even though the operating expenses and incremental revenue of this type of EVs offsets infrastructure investments, the truth is that it many users are still not convinced of making such investment. Furthermore, V2G implementation entails the study of charging station design, which in practice could signify dealing with time-consuming legislations. Thanks to these and many other concerns around V2G, researchers have dedicated efforts to find ways to test and prove their hypotheses.

2.2.2 Moving towards a scalable modular testbed for V2G implementation

If we can find a solution where Vehicle-to-Grid capabilities can be tested, optimized, and proven, in a low cost and risk environment, then the implementation and workaround of their concerns would direct us towards a renewable, efficient, and sustainable grid.

We envision that through the development of a scalable modular testbed, V2G vehicles will be tested and serve as a global and consistent source of renewable energy soon. Theoretically, if most mobile transportation methods adopt this technology, we would be able to decrease stress on the power grid and enhance the potential to achieve grid balance [21]. This, without the necessity of expanding power generation infrastructure. V2G offers the potential to provide cheap and fast decentralized energy storage. Hence, it is our environmental responsibility to optimize energy efficiency with such a possibility.
2.2.3 Implementing Autonomous Vehicles with V2G capabilities

Combating climate change, reducing energy cost for consumers, and energy conservation have always been efforts that affect all humans. However, the increase in power consumption every year combined with the lack of energy efficiency in the grid have made these efforts increasingly difficult to meet [22]. For this reason, many actors in the power generation sector have investigated diverse ways to achieve an efficient and stable grid.

Thankfully, the growth of electric vehicles and the smart grid led these actors to the creation of vehicle-to-grid technology. Additionally, Autonomous Vehicles (AVs) are another technology that is in the developing stages and that could have beneficial repercussions for energy efficiency. Autonomous vehicles have the potential of using significantly less energy when driving, compared to a vehicle driven by a human. According to the National Renewable Energy Laboratory [23], AVs could lead to congestion mitigation, efficient routing, less breaking, platooning, and lighter vehicles which would also make them more energy efficient. Thus, these solutions lead us to explore the benefits of energy efficiency when combining both technologies.

2.3 Proposed Solution

Ideally, these proposed autonomous electric vehicles would be able to detect their own energy levels, identify the nearest charging/discharging station accordingly to the situation, and navigate and dock autonomously to the specified destination in an urban environment. In reality, cities and towns are still not convinced to address the initiative even though that, according with the KPMG 2021 Global Automotive Executive Survey [24], more than 1,100 auto executives in 31 countries estimates 52% of new vehicle sales to be all-electric by 2030 in the U.S. Nevertheless, the implications of implementing this type of technology in an urban setting carries several security
risks worth considering. For this reason, it is required to conduct exhaustive testing in a controlled environment.

This project proposes to study and explore the implementation of V2G autonomous electric vehicles in urban environments by developing a small-scale platform that allows easier testing, research, and development of this technology. By getting inspiration from the well-known platform Duckietown [25], this work will be contributing and add value by incorporating pedestrian recognition and EV charging type capability to the mobile simulated environment. To make this possible, it is intended to make use of electric robots capable of autonomous docking and navigation. Some of the requirements for these robots include the ability to navigate the circuit while avoiding objects/pedestrians, follow the road by itself, and navigate and dock autonomously to their respective charge/discharge station when a command or state is triggered.

2.3.1 Scale up from testbed to on-campus implementation

The proposed platform is initially intended to be a portable representation of the UTEP campus. However, once proper testing, research, and development have been satisfactorily conducted, our vision is to scale up this initiative to implement autonomous transportation vehicles around the campus. This envisioned scale-up would constitute the next step to making UTEP a pioneer in the areas of Autonomous Navigation and Vehicle-to-Grid.

2.4 Research Questions

At the completion of this study, it is aimed to answer the following questions:

- How does the object recognition model testing accuracy compare between figurines, Legos, and real people's faces?
- How effective and safe can autonomous navigation be in a university-based pedestrian environment?
• What are the best methods in software and hardware to ensure a successful autonomous charging station docking?
• What impact and implications will self-driving cars have on road designs?

2.5 Chapter Summary and Transition to Autonomous Vehicles

In this chapter, the concept of V2G was covered in the first section. In it, the reader could understand the definition of it, its benefits, and existing barriers. It was explained how by adapting this technology, the world could benefit from stabilizing the grid and integrating renewable energies into a balanced energy system. Then, it was mentioned how a testbed could be beneficial to prove many theories without having to make a high investment. At the end, the motivation of implementing autonomous driving into the initiative is described as well as the high-level approach to make this possible.

In the next chapter, the subject matter will focus on the technical aspects of autonomous vehicles as it is the focal point of research and study of the work done in this thesis.
Chapter 3: Autonomous Vehicles

3.1 How do Electrical Vehicles work?

Before deepening into the concept of Autonomous Vehicles, it is necessary to understand the basic functionality and structure of EVs. Electric Vehicles (EVs), in contrast with contemporary cars, do not operate with an internal combustion engine. Instead, they are powered by an electric motor. This motor gets its energy from a controller that regulates the power given depending on the accelerator's position. The car's main source of power comes from a battery bank which can be recharged by connecting the EV to a power outlet [26].

Because electric cars do not use an internal combustion engine, they tend to produce fewer emissions, be more power efficient, and contribute to sound pollution by being silent. Besides this, an EV operates in a comparable way to an automatic car. The basic operational concept of EVs starts when the driver presses the accelerator. Once this happens, the DC power from the batteries is transformed into AC (Alternating Current) power. Then, the controller adjusts the AC frequency from the inverter to the motor. This would make the motor rotate accordingly and through a cog move the wheels. In addition, whenever the driver presses the brakes or decelerates, the motor becomes an alternator, producing DC power. This power is then sent back to the battery [27].

3.2 Autonomous Vehicles Background

3.2.1 Defining Autonomous Vehicles

Autonomous Vehicles (AVs) are an emerging technology in the transportation sector with great potential to change the way individuals and communities interact with the environment. A concrete definition of this technology varies as each definition is based on levels of automation. According to the Environmental and Energy Study Institute [28], these automation levels go from level 1 (Driver assistance) to level 3 (conditional driving automation), to level 5 (full driving automation). However, according to the most general and widely accepted definition across the
scientific community, an autonomous vehicle (AV) is a vehicle that can guide itself without human conduction [29]. This technology comes with as many promising advantages as barriers which are worth mentioning some of them.

**Advantages**

Firstly, advocates claim that autonomous vehicles will be significantly safer than vehicles driven by humans. According to the United States Department of Transportation (USDOT), 94% of vehicle accidents are down to human error [30]. This implies that the implementation of reliable artificial intelligence into these vehicles could save 90% of human lives in car accidents.

Fewer accidents will lead consecutively to less traffic and congestion which at the same time will mean a drop in emissions. Ohio University’s Future of Driving report [31] states that “optimized driving can cut emissions up to 60%, and driverless cars can be programmed to maximize these solutions”. In addition to this, according to the U.S. Department of Energy (DOE) could reduce energy consumption by as much as 90% if used properly [23].

It is hard to tell with so many implications and variables how the economy will be affected due to the implementation of autonomous vehicles. However, there have been many projections that make this initiative promising in the economic sector as well. According to an estimate by Intel Corporation in 2017 [32], the economic effects of autonomous vehicles will total $7,000 trillion in 2050. This projection assumes a new ‘passenger economy’ where consumers and businesses will adopt Mobility-as-a-Service (MaaS) as their business model. Besides this, autonomous vehicles expected to reduce and exchange unproductive driving hours for working hours boosting their productivity and with it the economy as well [33].
Barriers

Even though autonomous vehicles have promising implications towards a more energy efficient transportation system, there are also many concerns around the initiative that has not let AVs transition into our current world. For us, understanding these concerns is a crucial aspect of the development of our proposed platform.

There is no guarantee that technology will neglect accidents completely. In the last year, statistics released by U.S. safety regulators reported 400 crashes of vehicles with partially automated driver-assist systems [34]. This makes passengers have a false sense of security as accidents involving this technology are something out of their control. Hence the need to improve the safety rate among these vehicles.

Moreover, according to [35], the widespread of AVs will inevitably come with many changes in urban infrastructure. The cost of road infrastructure changes, maintenance, and new regulations could be something that will need robust testing and simulation before going through. For autonomous systems, roads may need to be kept free of any debris or damaged or irregular pavement that could affect the vehicle’s sensors and with it the safety of the passengers or pedestrians.

The autonomous vehicles definition encompasses a wide variety of approaches and sensors that can be used to make it possible. For this reason, it is important to expand into the key sensors for autonomous vehicles: camera, radar, and lidar [36].

3.3 Sensors in Autonomous Vehicles

As mentioned before, the most crucial components of autonomous vehicles are their sensors. They are used to collect data from the environment which then is analyzed by the computer of the vehicle to determine the corresponding speed, break, steer, and desired behavior. Hence, it is important to go over some of the most used sensors for autonomous navigation to determine which of them can be of use in the study.
3.3.1 Camera

Cameras were one of the first sensors used in AVs and until today they have been manufacturers' main choice. They are used for tasks related to seeing and interpreting objects on the road, just like humans do with their eyes. Its efficiency in texture interpretation, data classification, and the fact that they are widely available and more affordable than radar or lidar make the camera a good fit for AV implementation. Since the camera is an indispensable component in AVs, there has been a lot of research around applications and techniques. Applications around these sensors include road following, image classification, semantic segmentation, and among others [37-39]. There is no such thing as the perfect sensor. Looking into some drawbacks of the camera sensor it is important to mention that they are environment dependent. This means that rain, fog, snow, or even backlit can prevent these sensors from working properly, which at the same time would increase the likelihood of accidents [36]. Furthermore, there could be situations where the image taken from the camera is just not good enough to make a proper prediction.

3.3.2 Radar

Radar stands for Radio Detection and Ranging. These sensors have been around for a long time, and due to their capability of measuring speed directly thanks to the Doppler effect [36], they can be used for effective adaptive control, collision avoidance, among others. However, even though radars behave better in difficult weather conditions in comparison with cameras or lidars, they generate less angular accuracy and data than the other two. Besides this, according to [40], radars are propense to jamming attacks. This is when a signal generator is used to send out constant signals in the same frequency range of the actual MMW radars on the vehicles.
3.3.3 Lidar

Lidar stands for Light Detection and Ranging. They use infrared laser beams to determine the distance between the sensor and the surrounding object. Apart from this, lidar allows creating of 3D images of the detected objects and mapping of the surroundings which fulfill a very handy capability for AVs [36]. Unfortunately, these sensors are significantly more expensive than the other two sensors previously mentioned. In [41], the author mentions that there are two major problems with lidar. First, the prices for competent lidars that could handle autonomous navigation range between $10,000 to even $80,000. Added to this this sensor performance is weather dependent, which makes us decide to keep the use of lidar for the future.

3.4 Chapter summary and the use of the camera sensor in low-cost platforms

Even though these three sensors are key components of AVs, the practical implementation of some of them for the intended low-cost testbed does not fit with the anticipated requirements. By understanding that lidar and radar sensors would not be a feasible low-cost or reliable implementation, it was determined that the use and exploration of the camera sensor will be the best starting point for our study. Thankfully, the camera sensor is a versatile sensor that has been widely used in similar low-cost robot platforms [42-44], providing solutions in the areas of object recognition, road following, and human interaction. These versatile approaches confirm that with the use of just one camera, there is no need of using any other sensor to achieve the three main functions of our study: Road following, Object/Pedestrian recognition, and Autonomous docking.

The goal of this chapter was to give a general understanding of autonomous vehicles functionality. Besides, this chapter set the basis for the help us realize the promising benefits of developing autonomous vehicles with V2G capabilities. Yet, it also brought the need and motivation for a testbed where it could tackle the current concerns of these technologies by looking into a hybrid approach.
Chapter 4: State of the Art & Machine Learning Techniques

Now that the concepts of vehicle-to-grid and autonomous vehicles have been covered, and that the need of developing a low-cost platform for testing has been identified, it is appropriate to indagate over the state of the art in this matter. This chapter starts by exploring and comparing existing low-cost robotic platforms for autonomous navigation. The intention of this is to have a starting point that will lead us towards the development of Miner-town’s key features (road following, object/pedestrian recognition, and autonomous docking). The chapter ends with a literature review among machine learning techniques needed to achieve the features previously mentioned.

4.1 Simulated Environments

The proposed solution to the need for a controlled environment for testing, research, and development of V2G technology in automated vehicles led to search for alternatives where conditions could be as realistic as possible. Through this research, it was found that many autonomous navigation simulators are based on software. The problem with this is that software simulators do not always perform as expected when models are moved into a physical implementation. In the following sections, we will review two of the state-of-the-art platforms that serve as inspiration and foundation of the work done.

4.1.1 Duckietown

Duckietown is an open source, inexpensive, and flexible platform for autonomous education and research. Conceived as a graduate class at MIT in 2016 where a group of over 15 postdocs and 5 professors were involved [45]. Their goal was to build a small-scale platform that could preserve the scientific challenges of autonomous navigation.
As shown in Figure 4.1, the platform comprises small autonomous vehicles (“Duckiebots”) and cities (“Duckietowns”) and is completed with roads (constructed from exercise mats and tape), signage, traffic lights, obstacles, and citizens (duckies). Duckiebots sense the world with only one monocular camera and perform all the processing onboard with a Raspberry Pi. Duckietown is a useful tool since educators and researchers can save money and time by not having to develop all the necessary supporting infrastructure and capabilities. Many virtual simulators have been developed around autonomous vehicles implementation [46-48]. However, what differentiates Duckietown from the software base simulators is the capability of running physical tests at low
cost. For this reason, this platform design will take high inspiration into the Duckietown specifications mentioned in [25].

4.1.2 Jetbot AI Robot

Jetbot is an open-source robot platform which is powered and supported by the NVIDIA Jetson Nano AI computer. In contrast with the Duckiebots powered by a Raspberry-Pi, this platform is a small but powerful computer that is capable to support different sensors while running multiple neural networks in parallel for applications like image classification, object recognition, segmentation, collision avoidance, and road following. All of these while operating as low as 5 watts of power. Table 4.1 shows its technical specifications.

| **Table 4.1: Technical specifications for NVIDIA Jetson Nano Developer Kit** |
|---|---|
| **GPU** | 128-core Maxwell |
| **CPU** | Quad-core ARM A57 @ 1.43 GHz |
| **Memory** | 4 GB 64-bit LPDDR4 25.6 GB/s |
| **Storage** | microSD |
| **Video Encode** | 4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 |
| **Video Decode** | 4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 |
| **Camera** | 2x MIPI CSI- DPHY lanes |
| **OS** | Linux for Tegra (L4T) |
| **Display** | HDMI and display port |
| **Others** | GPIO, I2C, I2S, SPI, UART |

As Duckiebots, there have been many other low-cost robotics platforms capable of performing simple autonomous navigation tasks [49-51]. However, we want to make the platform as close as possible to the computational power of an EV. For this reason, and because of its
computational power, low cost, and versatility for artificial intelligence applications, we decided to make use of Jetbot for our study. In the next sections, we will review some of the most popular work and techniques used for autonomous navigation and automated robot docking. This review will constitute the basis of our work in the implementation process.

4.2 Machine Learning Techniques

It has been mentioned that that this study will focus its work on the development of three key features: Road/Line Following, Object/Pedestrian Detection, Autonomous Docking. Hence, we have decided to focus our research on techniques that could help achieve two specific goals: to detect and classify objects and pedestrians with 97%+ accuracy on the road, and to have a consistent road/line following behavior. By taking advantage of the Jetbot computational power, the following machine learning techniques will be used to integrate each one of our features.

4.2.1 Supervised Learning in Data Classification

Supervised learning is a technique used in machine learning that is characterized by using labeled datasets to train algorithms which in turn will classify the data or make predictions accurately. Back-error propagation is the most widely used supervised learning method in Artificial Neural Networks (ANN) paradigms. This method has been applied to pedestrian/object recognition, image classification, and a variety of pattern-analysis problems [52].

How does supervised learning work?

According with Dayhoff [52] back-propagation’s learning and outcome procedure is intuitively appealing because of its simple concept: “When the network is given an input, the updating of activation values propagates forward from the input layer, through each internal layer, to the output layer. The output units then provide the network’s response. When the network
corrects its internal parameters, the correction mechanism starts with the output units and back-propagates backwards, through each internal layer, to the input layer.”

At a high level, the supervised learning technique is divided into the training algorithm and the inference phase [53]. The training algorithm is the process of updating the weights of the ANN. At this stage, the desired output for each set of inputs is known and fed into the network. Then, the difference between the difference (error/loss) is calculated and used to modify the weights until the error is minimized. Figure 4.2 illustrates this process to a high level. Once the training algorithm has been optimized, the testing results are collected through feeding forward the trained network. This phase called inference, will not change the weights once the output is generated.

Figure 4.2: High level scheme of how supervised learning works. Fig modified from [54]
How effective can this approach be for pedestrian classification?

There have been many studies and proof that this supervised machine learning technique is highly efficient in classification problems. In [55], real-time supervised learning is used to train an autonomous vehicles classifier. By doing this, the author could provide an accuracy of over 90% in a range of 50 m. We have seen in the past how the use of Convolutional Neural Networks models trained through complex backpropagation methods has achieved up to 99% accuracy. An example of this is presented in [56], where the author uses an approach for pedestrian classification based on deep-learning (CNN) strategies using camera and LIDAR data. However, this accuracy normally depends on the complexity of the dataset and detecting pedestrians at any angle and the setting is something that must be taken into consideration. Thankfully, according to the study conducted by Billones [57], achieving a vehicle-pedestrian classifier with an average accuracy of 98% is totally feasible.

4.2.2 Transfer Learning for Visual Categorization

Transfer learning is a machine learning method where a model trained for a certain task is repurposed and reused as the starting point of another model with a different task [58]. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. For us, this technique will be especially useful for repurposing well known and highly trained models (like Alexnet or ResNet) into our smaller pedestrian and traffic signs classification model.
How transfer learning works?

Transfer learning is mostly used across computer vision applications due to the huge amount of computational power required. In these applications, neural networks normally try to detect edges in the earlier layers, then they detect shapes in the middle layers, and some feature specifications in the last layers. In transfer learning, the first and middle layers are used while just the last layers are retrained for the specific task that we like. In Figure 4.3, we can see that a neural network is pre-trained on ImageNet, then the trained weights of the first layers are transferred so they can be trained in the last layers to detect different disease classifications.

Why use Transfer Learning?

There are several benefits to using transfer learning. However, the main advantages of this technique that have been shown in academia results are, saving training time, better performance of neural networks, and reduced need of collecting a huge amount of data. With transfer learning, a model can be built with significative less training because the model is already pre-trained.

According to the analysis charts and tables from the study conducted in [60], the results show that transfer learning not only increased the accuracy of their classification model, but it also
gave insignificant overfitting. Also, in [61] it is shown that the application of transfer learning was a crucial part of pedestrian classification and identification. Thanks to it, they could reduce computational complexity and have been shown to have better performance than the models trained using traditional methods.

4.2.3 Data Splitting

It is often said that a machine learning model is as good as the quality of the data. One of the first decisions that must be made when working on modeling problems is how to use the existing data. Intuitively anyone would think that using all the available data to train the model would be the best way to proceed. And even doing this would make the machine ‘learn’ most features of our dataset, it would also have a challenging time generalizing the data. Overfitting is a concept in data science that happens when a model fits exactly against its training data. When this occurs, the model performs poorly over unseen data, defeating its purpose. To avoid this, researchers often split the data into training, verification, and testing sets [62].

The training set is the set of data that is used to train and make the model learn the hidden features/patterns in the data. In each epoch, the same training data is fed to the neural network architecture repeatedly, and the model continues to learn the features of the data. The validation set gives information that helps us tune the model’s hyperparameters and configurations accordingly. The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch (iteration). The main idea of splitting the dataset into a validation set is to prevent our model from overfitting, as if the validation loss is like the training loss, then we can conclude that there is no overfitting. Lastly, the testing set is a separate set of data used to test the model after completing the training. It provides an unbiased final model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "How well does the model perform?" [63].
What is the best split ratio?

The split ratio is dependent on the size of the data set. Many researchers have used the standard ratio 80/20 as a starting point and in the case that the data set is large some have gone into a 90/10 ratio. The truth is that there are many configurations, with many variables that affect the best split ratio. In [64], the author makes a comparative study of data splits across ML (Machine Learning) models for soil shear strength predictions. And based on the statistical analysis, a ratio of 70/30 for training and testing datasets was considered the best ratio for training and validating the models.

4.2.4 SoftMax activation function

The activation function in a neural network defines how the weighted sum of the inputs is transformed into a specific output. This function is what makes the network capable of learning and performing more complex tasks. These functions map the output between 0 to 1 or -1 to 1. There are many types of activation functions that are used for specific tasks. However, the nonlinear functions are differentiated by their range or curves. In this section, we will deep into the Sigmoid, ReLU, and SoftMax functions.

Sigmoid

This function is the most widely used activation function as it transforms the values into the range of 0 to 1. Therefore, it is specially used in applications that need as an output a probability. This function is defined as: \( \frac{1}{1+e^{-x}} \) [52].

ReLU

ReLU stands for rectified linear unit and is the most used activation function in the world by now. The reason for this is because it is used in all convolutional neural networks. Its range is from 0 to infinity so any negative values as an input are neglected. ReLU is more efficient than
other functions because certain neurons are activated at the same time. Mathematically it is defined as: \( f(x) = 0.01x, x < 0 \mid f(x) = x, x \geq 0 \) [65].

**SoftMax**

According to [65], the SoftMax function is a combination of multiple sigmoid functions. As we know, a sigmoid function returns values in the range 0 to 1. SoftMax function, unlike sigmoid functions which are used for binary classification, can be used for multiclass classification problems. The function, for every data point of all the individual classes, returns the probability. SoftMax is defined by the equation:

\[
P(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}.
\]

<table>
<thead>
<tr>
<th>Input pixels, x</th>
<th>Feedforward output, y</th>
<th>SoftMax output, S(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cat</td>
<td>Dog</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 4.4: SoftMax function implementation

In [65], a SoftMax activation function module was designed and implemented on FPGA. The implemented model could be configured to read any number of inputs up to 1024 elements. For our purpose this is a promising function to explore since our pedestrian/object recognition module will fall into a multiclass classification problem.
4.2.5 Cross-entropy loss function

In Artificial Neural Networks, the loss or optimization function is how the model takes the predicted value, compares it with the desired output, and from it modifies through some equation the weights of the ANN. Their objective in most of the cases is to minimize the loss and optimize the model during training. Cross-entropy is the most popular loss function for multiclass classification models; hence it normally goes hand in hand with the SoftMax activation function. The corresponding mathematical equation for the cross-entropy loss is defined as: \( L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) \); were \( t_i \) is the truth value and \( p_i \) is the SoftMax probability for the \( i^{th} \) class. The equation compares the distributions over the entire domain \( n \) of the \( i \) variable and it only assumes positive values.

$$ L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) $$

![Figure 4.5: Cross-Entropy simple schematic](image)

According to [66], cross-entropy is widely employed due to its fast computation. However, it only considers the true class probability \( p_i \) without considering the mass distribution among the other classes.
4.2.6 MSE loss function

Mean Squared Error (MSE) is the most common loss function used in regression models. The loss is the mean data in observance of the squared differences between true and predicted values. This is better described as the formula: \( L = \frac{1}{N} \sum_{i=0}^{N} (y - \hat{y}_i)^2 \). The MSE loss function because of the squaring part of the function, puts larger weight into big errors while small errors are less penalized. This behavior is good for ensuring that the trained model has no outlier predictions with huge errors.

4.2.7 Image-Coordinate Probabilistic Regression

Regression analysis is the process of estimating the relationship between a dependent variable (output) and independent variables (inputs) from labeled training data. Using regression models, a function is fitted to the available data and based on the error obtained, the function is adjusted to obtain the desired output. It can then be capable of forecasting trends or predicting precise outputs from unseen data. One important consideration to take when working with regression models is the quality of the dataset. Care must be taken to select representative and diverse data from the overall population. If the training data is not representative, then there is the risk that the predictions will not be accurate enough.

Applications of regression models in computer vision

Regression has been a very popular approach in different computer vision applications. In [67] the author deploys a CNN (Convolutional Neural Networks) for robot perception which was trained on a custom dataset. This data set was then fitted into a regression model that predicts the real distance-to-collision in view of the robot’s camera input. Another example is the work done by [68], where the authors present a Multi-Task Regression-based Learning (MTRL) approach that allows a UAV (Unmanned Aerial Vehicles) to perceive obstacle-free areas while simultaneously predicting the orientation required to explore the environment safely.
These approaches inspire us to create a regression model capable of estimating the relationship between what the robot sees and the targeted direction to go. This can be very useful for predicting a target coordinate from an input image where applications on road following and robot calibrations are over the scope of work.

4.2.8 Automated robot docking

Implementing V2G technology into autonomous vehicles entails the development of an automated docking feature. Research around this capability is related to drone autonomous dynamic landing, autonomous latching systems, indoor navigation, autonomous parking, among others [69-72]. Inspired by these studies we decided to make use of a line following approach.

Line Following Algorithm

Another technique worth considering simplifying the autonomous docking system is the widely known line following algorithm. In [73] a proposed solution for smart autonomous parking through line following techniques is presented showing positive results. However, a variety of sensors, and many hardware and software resources had to be used to perform the intended task. In other studies [74-76], the authors acknowledge this situation and develop a camera-based line following the prototype. By applying data collection procedures, image processing techniques like filtering and color thresholding, convolutional neural networks, and Proportional-Integrate-Derivative (PID) control the model was able to effectively replace a human operator for a single task.
These approaches encouraged us to experiment and research line following as an assistance technique for a robust and reliable autonomous docking system. Furthermore, we are eager to see how the implementation of line following mechanisms can improve this feature.

4.3 Pre-trained models

When training a neural network for classification or regression, the first layers of the networks are just able to detect simple features. As layers are added into the network, more sophisticated features like shapes or patterns can be identified. Finally, the last layers of the network are the ones that can classify faces, objects, or different classes. This can be possible since the weights of the model are already set for classification. Yet, because the first layers of these neural networks will always identify lines or shapes, it is not time efficient to train these layers each time that a neural network is created. Hence, the use of pre-trained models is used to speed up the training process. In this section, two of the most popular pre-trained models in the computer vision domain will be discussed to decide the most fitted for this study [77].
4.3.1 Alexnet

Alexnet is the leading architecture of pre-trained models in the computer vision domain. According, with [78], this deep convolutional neural network was developed out from the need of improving the ImageNet challenge (which has almost 14 million images across a thousand classes). The neural network architecture, which has more than 60,000 parameters, consists of 5 convolutional layers plus 3 fully connected layers with a final 1000-way Softmax [79]. Additionally, by using the non-saturating ReLU activation function, the training performance improved. With this, Alexnet has become a powerful model capable of achieving high accuracies on a wide variety of datasets. However, adding or removing any of the convolutional layers provoke a decrement on Alexnet’s performance. In 2012, Alexnet won the ImageNet competition with a top-5 error rate of 15.3%.

4.3.2 Resnet

After the impressive results that Alexnet showed in 2012, Residual Network (ResNet) is the most innovative work in the last years [80]. ResNet main idea was the introduction of identity shortcut connections that skips one or more layers. These connections allowed to train from hundreds to thousands of layers without affecting its performance. This behavior let the network to tackle the vanishing gradient problem which entails that as the layers of the network increases, its performance decreases. In 2015, a residual neural network was used to win the ImageNet competition, and from then it became the most cited neural network of the 21st century [81].
4.3.3 Pre-trained models’ comparison

In Table 4.2, the comparison between Alexnet’s and Resnet’s top-5 accuracy, their number of trainable parameters, and the floating-point operations (FLOP) needed to do a feed forward pass are presented. It can be observed that both architectures offer around 60M training parameters. The most notable advantage of Resnet-152 is the accuracy rise of almost 10%. However, this model also requires about 10 times more computational power compared with Alexnet’s FLOPs, which translates to more training time and energy needed. For this reason, this study will make use of the Alexnet’s pre-trained model alongside the transfer learning technique to design the desired neural network.

<table>
<thead>
<tr>
<th>Network</th>
<th>Year</th>
<th>Top-5 accuracy</th>
<th>Parameters</th>
<th>FLOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexnet</td>
<td>2012</td>
<td>84.70%</td>
<td>62M</td>
<td>1.5B</td>
</tr>
<tr>
<td>Resnet-152</td>
<td>2015</td>
<td>95.1%</td>
<td>60.3M</td>
<td>11B</td>
</tr>
</tbody>
</table>

4.4 Chapter summary

In this chapter the state of the art of low-cost robotic platforms is described. From this analysis, it is decided that Duckietown and the NVIDIA Jetbot would be the optimal platforms to achieve the best possible physical simulation of autonomous vehicles for V2G integration under realistic circumstances. After this, the key features of the platform are defined (Road/Line Following, Object/Pedestrian Classification, and Autonomous Docking). Subsequently, we go through a literature review of the Machine Learning techniques intended to use to make these features possible. Finally, this chapter present and compare two of the most popular pre-trained models in the computer vision domain.
Chapter 5: Research Design and Methodology

5.1 AIMS AND OBJECTIVES

V2G technology will become the future of transportation and energy regulation. It is also clear that autonomous vehicles will be a technology that will eventually become a reality. Every year, EVs are disrupting the market in a positive way, and it will be just matter of time to start observing these technologies introducing us into the next energy revolution. However, as we learned in the previous sections, there are still many questions and obstacles to solve before implementing these innovative technologies in a civilian setting. For this reason, the objective of this research is to create a test-bench through experimental simulation where many aspects and problems of both could be solved.

5.2 APPROACH

The proposal is simple, to create a scale model of the University of Texas at El Paso where small electric robots could help us simulate, test and experiment many of the research gaps around V2G technology. The electric robot will fulfill three main functions: Navigate autonomously through road following techniques, detect, and classify different common objects and pedestrians in the road, and autonomously dock into the charging station.

Structure

The research is structured into three modules. The first module is about road following. In this module the robot should learn and improve its autonomous navigation system by just using the camera. Techniques like supervised learning and regression will be used to make this feasible.

The following module covers object/pedestrian classification. In this task, the robot must be able to recognize the campus limits, common transit signs and actuate accordingly with each
situation. Besides this, the robot should be able to detect, avoid, and to not injure any pedestrian under any circumstances.

The final module goes over autonomous docking. For this feature, we look to make a performance analysis within a machine learning line following algorithm. An important requirement for this module is to just make use of the camera. Figure 5.1 gives a visual representation of our research structure.

![Diagram of Miner-town modules and tasks](image)

**Figure 5.1: Miner-town modules and tasks**

### 5.3 Simulated Environment Design

#### 5.3.1 UTEP Campus Inspiration

One of our work’s visions is for Miner-town to scale up into a campus-friendly environment. For this reason, we have decided to base our road layout in accordance with a sector of UTEP’s campus.
Figure 5.2 sketches the proposed circuit road that our platform will be based on. The red closed circuit represents the road in which the robots will be operating; while the blue lines represent the sectors where the charging stations will be located.

5.3.2 Road Materials

To simulate the proposed university-based environment we investigate materials that could be portable, accessible, and affordable for future replications. For these reasons, we decided to take as reference the materials used in Duckietown [25].
Interlocking black tiles

61x61cm exercise mats are used to simulate the road. There are 3 different tile types that will let us create any desired road design.

![Road tile types](image)

- a) Straight
- b) Curve
- c) 3-way intersection

Figure 5.3: Road tile types

Notice how the use of duct tape is essential for creating each tile type. White tape represents the road limits. 5x5 cm yellow tape symbolizes the center mark for a two-way road. Finally, red tape represents an intersection in the road. The measurements used have been taking the closest as possible to the dimensions used inside the UTEP campus.

Waveshare Jetbot AI Kit

Duckietown used raspberry pi as its robot’s computer power. However, we decided to use the Waveshare Jetbot AI Kit since this platform is powered with the Nvidia Jetson Nano AI computer which can run multiple sensors and neural networks models in parallel.
5.3.3 Road Design

Inspired by UTEP campus, we decided to implement a close-loop road design in our platform. This design also has two 3-way intersection tiles, and a charging station tile in the middle of the road.

![Figure 5.4: Miner-town road design](image)

5.3.4 Bill of Materials

To make this project possible, there are some essential tools needed for the construction of Miner-town. The list of these tools alongside their potential cost can be found in Appendix D.

5.4 Chapter Summary

In this chapter, it is described in high-level, the methodology and materials to use for the study. Next chapters will focus on each of the modules presented in the research structure.
Chapter 6: Road Following Module

6.1 Objective

In this module, we will be collecting an image regression dataset that will help the Nvidia JetBot to follow the road. We will teach the JetBot to store a target x, y image coordinate and actuate accordingly to chase that target.

6.2 Approach and Framework

This module will be divided into the data collection section, training and validation section, and inference section. By using a neural network model and techniques like supervised learning, the robot will be trained to solve a regression problem. First, we will be storing road images in different angles. Each image will be stored (labeled) with the x,y coordinate in which we would want the robot to go. Once the dataset is created, the images are transformed into tensor form for proper processing. Afterwards, the model inherits the Resnet’s pre-trained model features through transfer learning and adjusts its last layers to redefine the model’s purpose to coordinate prediction. Finally, the data is split in a 70:30 ratio for training and validation, and through the MSE loss function the training process takes place.

Figure 6.1: Road following Framework
6.3 Dataset Creation

By using a gamepad controller, the robot could store the target coordinate according to the presented image. It is important to mention that the robot’s displayed image is a 224x224 pixel image mapped into the range of [-1:1]. What this means is that the center of the image will always be 112x112 in pixel form or [0,0] in the mapped coordinate system. In Figure 6.2, images from the data collection process are shown alongside the respective coordinates. The red circle in the images represents the current location of the robot, while the green dot represents the target coordinate to where we want to go.

![Figure 6.2: Road following data collection process](image)

6.4 Training and Verification Results

As mentioned before, in this section we trained a neural network to take an input image, and output a set of x,y values corresponding to the target direction. By using transfer learning, we repurpose the ResNet18 pre-trained model for a road following application. And with a dataset of 136 images and a 70/30 ratio, the MSE loss went from 1.00 to 0.02 in 50 epochs. It is also good to notice that training the model for 300 epochs did not offer significant changes. In Figure 6.3, it can observe the convergence between training and validation results with 50 and 300 epochs. In the model where 50 epochs were used the training loss was 0.0138, while the validation loss was
0.0414. Since the validation loss did not increase after 300 epochs it is safe to say that our model is robust, not overfitting and has “learned”.

![Training and validation graph results for road following](image)

**Figure 6.3: Training and validation graph results for road following**

### 6.5 Inference Results

In this section we run the trained model and feed it from different angles from the road. Our expectation is that the model would predict the coordinates to pursue based on the live images displayed by the camera. Figure 6.4 shows the results given by the model under 4 different scenarios (curve, right, center, left). It is important to remember that the image is mapped in [-1:1] model where -1 corresponds to the left or bottom edge of the image, 1 to the left or top of it and making the center of any image [0,0].

![Inference results](image)

**xy = (-0.64,0.07)  xy = (-0.93,0.16)  xy = (0.07,0.20)  xy = (0.67,0.17)**

**Figure 6.4: Road Following inference results**
These results confirm that the road following model works. The predicted coordinates correspond to where the robot is supposed to go. Finally, we make use of three equations to correlate the predicted coordinates with the robot’s steering value. These equations go as follows:

- \( \text{Angle} = \arctan2(x,y) \)
- \( \text{PID} = \text{angle} \times \text{steering gain} + (\text{angle} - \text{angle last}) \times \text{steering dgain} \)
- \( \text{Steering} = \text{PID} + \text{steering bias} \)

The convincing results lead us to think that our model and approach could successfully be used by anyone with an interest in self-navigation.

### 6.6 Odometry Constrains

Unfortunately, the open loop motors included with the Jetbot kit made the tuning process and the road following task inconsistent and unreliable. This behavior can be observed by running a simple command with the Jetbot. This task consists to drive the robot forward 50 times for 2 seconds. In Figure 6.5 it can be seen how the distance traveled, and the end point relative to its starting point varies from run to run. One solution to this problematic includes adding rotational encoders into the motors to enable odometry techniques. This will let us have better control of our motors, making the speed and navigation more precise and consistent.
In this chapter, the first module of the study is presented. The idea behind road following is for the robot to learn a specific circuit and traverse it without any human interaction. The approach used to make this possible was through supervised learning. By positioning the robot in every angle of the road, the user could store the desired coordinate to which the robot should go in the given scenario. Next, techniques like supervised learning and artificial neural networks were used to train the robot. Training and validation results conversion bring us to conclude that the model behaved properly, discarding any sign of overfitting. Finally, the inference process confirmed the correct prediction of x,y coordinates.

Figure 6.5: Robot behavior with open-loop motors

6.7 Chapter summary
Chapter 7: Object/Pedestrian Recognition

7.1 OBJECTIVE

The objective of this module is to give autonomous vehicles (Jetbot), the capability of recognizing and categorizing pedestrians and traffic signs with a prominent level of precision. As mentioned in the previous chapters, our approach in this module is to implement a multiclass classification model. There are 5 categories (classes) in which the robot will base itself to take decisions. These classes consist of pedestrian, stop sign, speed limit signs, parking signs, and free.

7.2 CLASSIFICATION CATEGORIES

7.2.1 Pedestrians

To simulate pedestrians, we decided to use 7cm – Fisher-Price figurines, and 3cm – Lego figurines. One important aspect of the selection process of these figurines is the variance in size and demographics.

Figure 7.1: Figurines used to simulate pedestrians.

We also extended our pedestrian model dataset by adding to the Fisher-Price figurines real people's faces. By doing this we can run tests and observe how the pedestrian detection model behaves under different circumstances.
7.2.2 Traffic Signs

Besides pedestrians, our model is intended to identify three essential traffic signs; stop, speed limit, and parking. We decided to use 9 cm – wooden traffic signs to simulate these traffic signs for our platform.
### 7.3 Module Approach and Framework

Our approach is divided into 3 sections (Data Collection, Training and Validation, and Inference). In the data collection section, first, it is needed to create the dataset by capturing and labeling the training data. It is important to mention that variance in lighting, angles, and samples are key for proper learning. We do this by manually placing the robot in different scenarios, taking pictures of them, and saving the images in their respective directory.

Once the dataset is completed, we take it into the training section. In this section, we first make use of some image processing transforms and techniques to feed our neural network with the dataset tensor form. Afterward, the model is designed by combining a backpropagation supervised learning process, and the Alexnet pre-trained model. The data is split in a 70/30 ratio to run the training process. Finally, in the inference section the model is executed with unseen data. By making use of the SoftMax activation function, we obtain the most probable classification of the respective input.

![Figure 7.4: Object/Pedestrian classification module framework](image-url)
7.4 Data Collection

The idea behind data collection for supervised learning is to place the robot in different scenarios where the 5 classification categories were present. It is important to mention that variance of angles, light conditions, and scenarios are fundamental to making our model robust and avoiding overfitting. For testing purposes, we first created two models with 50, 100 and 150 samples per category, respectively. Some samples of the data collection procedure can be seen in Figure 7.5.

Figure 7.5: Data collection process with 125 images per class.

7.5 Training & Verification

For the training and verification section our goal is to have our accuracy close to 1 and our CE (Cross-Entropy) Loss near to 0. To make this possible, first the available dataset is transformed into tensor form. Then, the dataset is split in a 70/30 ratio (70% for training, and 30% for validation). Next, the neural network is defined to be trained. By using transfer learning, we repurpose the pre-trained AlexNet model for a new task (pedestrian and traffic signs recognition) with much less data available. Thanks to this approach, we could reduce the resources and training
iterations (epochs) needed in the learning process. Finally, the neural network is trained in different configurations (number of epochs, and number of samples) to look for the best performing model.

7.5.1 Results

After running the training and validation process under different configurations, it was observed that a bigger number of epochs improves the accuracy and CE loss of the model. However, we want to be cautious and not increase too much the number of epochs. As the number of epochs increases, a greater number of times the weights of the network are changed and the curve goes from underfitting, to optimal, to overfitting. One crucial indicator when interpreting the results is by understanding the training and validation error. If the training error is low but the validation error increases, it indicates that the model is overfitting. In contrast, an optimal performance would be when both errors are low and mostly equal. Figure 7.6 and 7.7, show us a visual representation of how the number of epochs improves the final performance of the model.

![Figure 7.6: Training and Verification with 50 samples, and 30 epochs](image)
Next, in Figure 7.8 and 7.9 the number of samples increases from 50 to 100 to 150. The results let us observe that the number of samples per class on the training and validation dataset is directly proportional to the performance of the model. Simultaneously, it is perceived that as the number of samples increases, less epochs are needed to obtain suitable results. Finally, Table 7.1 demonstrates the quantifiable testing and verification results from different samples per class.
Table 7.1: CE Loss and Accuracy from the Training and Verification process

<table>
<thead>
<tr>
<th>Samples per class</th>
<th>Number of Epochs</th>
<th>CE Loss (T/V)</th>
<th>Accuracy (T/V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>30</td>
<td>0.000725</td>
<td>0.98</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>0.0009</td>
<td>1.00</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>0.0005</td>
<td>1.00</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0.0006</td>
<td>1.00</td>
</tr>
<tr>
<td>150</td>
<td>30</td>
<td>0.0003</td>
<td>1.00</td>
</tr>
<tr>
<td>150</td>
<td>100</td>
<td>0.0001</td>
<td>1.00</td>
</tr>
</tbody>
</table>

7.6 Inference Results

In the inference section, we ran our model across the 5 defined classes: Pedestrian, stop sign, speed limit sign, parking sign, and free. Our objective is to record the performance of our model in each class and determine potential limitations or improvement opportunities within our trained model.
7.6.1 Pedestrians Accuracy

There are three different representations of pedestrian in our platform (Fisher-price figurines, Legos, and figurines with real people’s faces). For measurement parameters we decided to place these pedestrians in 6 scenarios relative to the robot’s camera: Front with intersection, side with intersection, front in the road, side in the road, fall, and backlit.

![Pedestrian detection results graph](image1)

Figure 7.10: Pedestrian detection results graph

![Pedestrian detection results with backlit](image2)

Figure 7.11: Pedestrian detection results with backlit
Looking into the results in Figures 10-13, we can appreciate that one major limitation of our model is whenever a pedestrian is under backlit. Backlit accuracy is between 75% to 90%. However, the most concerning metric for the backlit scenario is its recall. Recall is defined as: $$\frac{TP}{TP + FN}$$ and it is used whenever there is a high cost associated with false negatives (Detecting a pedestrian as free to go). In our model, the veterinarian figurine in backlit conditions would just be detected 86% of the time or 43 out of 50 times.

On the other hand, encouraging results were obtained from any other scenario without backlit. With an average accuracy of 99.64% among every class, it is proven that our approach and model work well detecting pedestrians of varied sizes and from any demographic, even if the
figurines were covered with real people faces. In Table 7.2, a more detailed report is presented from each pedestrian category and its scenario accuracy.

Table 7.2: Testing accuracy of pedestrian recognition model.

<table>
<thead>
<tr>
<th></th>
<th>Front</th>
<th>Side</th>
<th>Fall</th>
<th>Front (Int)</th>
<th>Side (Int)</th>
<th>Backlit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Veterinarian</td>
<td>100%</td>
<td>99.96%</td>
<td>99.93%</td>
<td>99.99%</td>
<td>99.99%</td>
<td>77.99%</td>
</tr>
<tr>
<td>Police</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>99.92%</td>
<td>99.99%</td>
<td>99.98%</td>
</tr>
<tr>
<td>Fire-Fighter</td>
<td>99.99%</td>
<td>99.99%</td>
<td>100%</td>
<td>99.02%</td>
<td>99.99%</td>
<td>89.15%</td>
</tr>
<tr>
<td>Doctor</td>
<td>99.98%</td>
<td>99.97%</td>
<td>99.86%</td>
<td>99.33%</td>
<td>90.45%</td>
<td>92.45%</td>
</tr>
<tr>
<td>Teacher</td>
<td>100%</td>
<td>99.98%</td>
<td>100%</td>
<td>99.99%</td>
<td>99.72%</td>
<td>94.12%</td>
</tr>
<tr>
<td>Lego</td>
<td>100%</td>
<td>99.93%</td>
<td>99.96%</td>
<td>100%</td>
<td>99.99%</td>
<td>74.96%</td>
</tr>
<tr>
<td>Real Face</td>
<td>99.99% (FP)</td>
<td>-</td>
<td>-</td>
<td>100% (FP)</td>
<td>-</td>
<td>92.73% (FP)</td>
</tr>
<tr>
<td></td>
<td>99.91% (Lego)</td>
<td>-</td>
<td>-</td>
<td>99.90% (Lego)</td>
<td>-</td>
<td>75.19% (Lego)</td>
</tr>
</tbody>
</table>

7.6.2 Object Classification Accuracy

Besides detecting pedestrians, our model was trained to identify free roads, stops, speed limit, and parking signs. The scenarios are like the ones used on pedestrians; just including the parking area and curve road for the park and free classes. In this section, we fed every class to our model 50 times per scenario.

For the free class, the accuracy in every scenario was on average 97.57%. There were just 2 scenarios in which the model underperformed. With a recall of 96% and average accuracy of 94.41% the robot miss classified 2 out of 50 samples at intersections and parking area. Fortunately, for this class in specific, false negatives are not as critical as in other classes. It was also noticed that the stop sign got an average accuracy of 99.92% across each scenario. This was expected as the shape and color of the object is unique to anything around the environment. The two last objects presented a challenge as the shape of the figures were similar. It was not a surprise that under backlit conditions the classification accuracy would drop for both classes. Similarly, there was a clear confusion whenever the speed limit sign was near the white line road limits. Figure 7.14 and Table 7.3 present a detailed representation of these results.
One special mention must be made of the case with multiple objects in a single model. Object detection is conformed of image classification and image localization. In this model we focus on creating a classification model so the robot could differentiate between pedestrians and traffic signs. However, without image localization, the model would struggle with multiple objects in the same image. The top part of Figure 7.15 demonstrates this phenomenon as pedestrians and a stop sign are introduced into the same image. In this case, the chart shows that our model can only predict one object at a time (values near 1 corresponds to pedestrian prediction while values
near 0 represents a stop sign prediction). In the other hand, the model behaves well with multiple objects of the same class in one image. Its behavior can be seen at the bottom of the figure.

![Image of pedestrian and stop sign](image1.png)

![Image of multiple pedestrians walking](image2.png)

Figure 7.15: Multiclass vs single class recognition

Since the classification model struggles when there is more than 1 object in the live image, there is the need to implement object localization feature into the model as well as image segmentation in future work to make the module more practical.

7.7 Chapter Summary

This chapter focuses its efforts on the object/pedestrian classification problem. By building a robust data set for each of the categories, the study could demonstrate that through supervised learning, the model would give positive results for classification. Limitations in the module includes: the presence of backlit in an image could affect the model reliability. Besides, it would be needed for future work to implement image segmentation so multiple classes can be detected.
Chapter 8: Autonomous Docking Module

8.1 Approach and Framework

There is no existing solution for autonomous docking in Duckietown. The goal of this module is to implement such functionality so potentially the robots or autonomous vehicles could charge themselves automatically. To realize this goal, we assume that the autonomous navigation module is functioning properly.

Our approach starts every time our robot detects an intersection. Once this happens, the object recognition module will look to detect a parking sign. At this point if the parking sign is detected, then the “entering parking lot” algorithm will be triggered. Otherwise, the robot will just keep its original path. Detecting another intersection at the entrance of the parking lot will trigger the line following and/or Apriltag algorithm. Our line following algorithm will have the same approach as the road following algorithm by training a neural network to identify the blue line which is aligned to the charging station connector. Additionally, the use of Apriltag detection algorithms will be explored to dynamically have awareness of the robot’s distance and position relative to the charging station. It is expected that this will help the robot calibrate and serve as complement of the line following algorithm. Figure 8.1 illustrates our approach and logic.

Figure 8.1: Autonomous Docking Framework
8.2 Parking Lot Design

Figure 8.2 depicts the proposed design for the parking lot where our charging station will be located. This parking lot design includes two intersections (one for entrance and one for exit), one blue dashed line aligned with the charging station, and continuous yellow tape in the perimeter to specify that we are in the parking area. The starting point (green dot) for this approach will always be the red intersection that is closer to the parking exit.

![Parking Design Layout](image)

Figure 8.2: Parking Design Layout

It is worth mentioning that the selected design was chosen based on practicality for testing. However, the potential variations for parking designs presents further research gaps and opportunities.
8.3 Charging Station Design

One of the main challenges around autonomous docking is the high level of precision needed to achieve a successful latching system. For this reason, a robust design for the charging station that facilitates this process is needed. There is just one requirement for the charging station design. To demonstrate any kind of energy transfer between the Jetbot and the charging station. It was decided to use the following materials to create the proposed charging station:

- 2x USB-micro magnetic connectors
- 1x ESP32 devkit
- 1x NeoPixel stick
- 1x Breadboard & 3x jumper cables

First, the USB-micro magnetic connectors are used for energy transfer. This works perfectly because the magnetic field created between the connectors lets the robot have some margin for error. One side of the connector is installed into the robot while the other side of the connector would be connected to an ESP32 devkit. Whenever a connection exists, the power of the robot is transferred to the ESP32, which consequently would turn on the installed and programmed Adafruit NeoPixel stick. Figure 8.3 shows the charging station design including the ESP32 schematic. In Figure 8.4, the energy transfer between the Jetbot and the charging station is demonstrated.
8.4 RESULTS AND PERFORMANCE

8.4.1 Parking sign recognition

As seen in the object recognition module, parking sign detection ranges between 65% to 99% of accuracy depending on the scenario and lighting conditions. Because detecting the parking sign is the only indicator available for triggering the autonomous docking algorithm it is important to understand the current performance of it. For this reason, we capture the recognition results of the parking sign when it is at the side of the first and second intersection. Figure 8.5 portrays the behavior of the parking sign recognition when backlit is available or not, and when the sign is in the parking lot intersection. Results show that with backlit the average accuracy for recognition drops to 65%. This happens because the model gets confused between parking sign (values near 1) and speed limit sign (values near 0). Without backlit, the average accuracy increases up to 93% while the model still confuses the parking sign with the stop signs. Finally, in the scenario that the parking sign is inside the parking lot, the average accuracy increases to 99%.
Figure 8.5: Parking recognition performance in different scenarios
8.4.2 Entering Parking Lot

For the entering parking lot algorithm, we assume that the parking sign was detected to initialize the process and that the robot starting position is over an intersection. From that starting position we have hardcoded the behavior of the robot to stop → go forward for 1.5 seconds → turn left 90 degrees → go forward for 2 seconds → stop. Figure 7.3 shows a simple code that was tested to simulate such behavior.

```python
# Move forward
for xabs in range(5):
    robot.set_motors(-0.36, -0.33)
    time.sleep(0.275)
robot.stop()
time.sleep(2)

# Turn Left
for x in range(5):
    robot.set_motors(0.00, -0.335)
    time.sleep(0.3)
robot.stop()
time.sleep(2)

# Move forward
for x in range(5):
    robot.set_motors(-0.36, -0.33)
    time.sleep(0.35)
robot.stop()
```

Figure 8.6: Tested code for entering the parking lot.

Unfortunately, due to the lack of encoders and consistency in our open loop motors, the robot had different output behavior every time the same code is executed. By referring to Figure 8.6, under the same circumstances the robot ended in different angles relative to the charging station. For future work, it will be crucial to add motors with encoders and employ odometry techniques into the existing robot to do proper testing of the algorithms.
8.4.3 Line following

After entering the parking lot, we assume that the robot will not be always centered towards the charging station or in range to detect the Apriltag. For these reasons we decided to implement a similar approach as the road following algorithm to train our robot to follow a blue line. As seen in the parking lot design, a blue dashed line is placed in-between the entrance intersection and the charging station.

The data collection, training and validation, and inference framework is used to accomplish this task. Looking into Figure 8.7, we can observe the convergence between training and validation, which proves that the model behaves accordingly. Notice that we use less samples to train this model since we are just interested in covering a small section of the road.

![Figure 8.7: Line following data collection and training and validation results.](image)

It is important to know that this algorithm is intended to serve as an alignment tool and stop once another fiducial marker is detected. Nevertheless, regression helps to redirect our image into the desired area of view.
Ideal conditions results

In the road following section, it was explained how the lack of odometry affected the overall precision of the physical motors in relation with the expected results. Consequently, this precision deficiency brought some constrains into the practical performance of our model. Nevertheless, the model was tested in ideal conditions to identify error deviation and set the base for improvement opportunities. These ideal conditions represent that the robot must be aligned with the docking station in a straight line at a 30 cm distance. After testing the model behavior over 50 times, the results in Figure 8.8 conclude that just 40% of the time a successful docking is achieved. Then, Figure 8.9 presents the error in cm relative between the magnetic connectors.

Figure 8.8: Autonomous docking success rate in ideal conditions

Figure 8.9: Error in cm at unsuccessful docking
8.5 Possible Improvement

In our closing section, we cover how the implementation of Apriltags can be used for motor calibration and successful docking. As mentioned in the idea, the robot detects the centered coordinate of the marker relative to the camera’s position and aligns itself accordingly towards that target. The approach for this task is divided into: Apriltag detection → robot alignment & docking.

AprilTag fiducial marker

Visual fiducial systems are artificial markers designed to be easily recognizable around an image. These fiducial systems goals and applications diverge from the intuitive relation with other 2D barcodes. For example, in contrast with the high information payload that QR codes require (hundreds of bytes), a visual fiducial operates only with 5-36 bits. This characteristic enables the capability of detecting the marker even if the image has: low resolution, is rotated in any direction, or is in unbalanced lighting conditions. AprilTag shown in Figure 8.10, is a visual fiducial system proposed and explained by E. Olson in [82]. As of today, these tags have been the most popular standard among the robotics research community. The AprilTag detection software computes the precise 3D position of the marker, its orientation, and identity relative to the camera. This made us decide to use AprilTag as a promising solution to the autonomous docking problem.

Figure 8.10: Examples of AprilTag families
E. Olson alongside his APRIL robotics laboratory team, developed the libraries and repositories for Apriltags detector. However, detecting Apriltags within Jetbot needed small modifications and tweaks to adapt this feature. After setting up the NVIDIA camera source and passing it as an argument, and defining the tag family type to be identified, the detector could recognize any Apriltag form the 36h11 family within a range of 14 inches. By getting the coordinates from the center, we could correlate the desired x,y values with the motors behavior and align the robot with the charging station for docking. It is worth recalling that due to the lack of encoder and odometry techniques the practical application lacks consistency. Yet, the theoretical results behind our methodology let us conclude that autonomous docking is feasible under these conditions.

![Apriltag detection and XY coordinates of center](image)

**Figure 8.11 Apriltag detection and XY coordinates of center**

### 8.6 Chapter Summary

This chapter covers the last of the three modules of this study. Autonomous docking is a feature not previously seen in Duckietown; however, our approach through line following and April tag implementation set the basis for a reliable and fully autonomous docking feature.
Conclusion

The recent evolution and projection of electric vehicles around the world have led to the development of innovative technologies. Thanks to this, autonomous navigation and vehicle-to-grid are now ideas worth considering. We have found that many advantages come around the implementation of both technologies, however there is a need to develop a controlled environment for research, testing, and validation of autonomous vehicles with vehicle to grid capabilities. In this study it is proposed the development of a testbed for autonomous navigation inspired on the simulation environment and platform Duckietown and the Nvidia Jetbot. Its novelty includes studies in pedestrian/object recognition, road following, and autonomous docking.

In this thesis it was first explained the motivation and purpose behind this study. It was explained how is that this study can not only benefit operators, investors, or automotive industries, but it could also be the foundation of future work around UTEP for pioneering in the areas of artificial intelligence, autonomous vehicles, control systems, and power infrastructure research (vehicle-to-grid). Next, a literature review was conducted on the most common techniques for object recognition, road following, and autonomous docking. Finally, we explained our approach for each of the modules and presented promising theoretical results for future implementation.

The results gave us a better understanding of the feasibility and limitations in each one of the modules. Positively, we find out that pedestrian recognition can be reliable with the approach and dataset collected. Limitations around this module included its fragility against backlit, and that our classification model only worked when there was just one class in the camera range. The road following section also threw encouraging results. By running the trained model in the inference phase, we got in return the desired target coordinate as an output. Unfortunately, the lack of encoders in the motors and odometry techniques prevented us from a consistent behavior in the practice. This could also be said by the autonomous docking module as precision and consistency was a crucial part of its practical implementation. Yet, the theoretical results from the Apriltag detector and line following model make us confirm that this can be solved in future work.
Future Work

This research has set the basis for the development of a testbed for autonomous vehicles with V2G capabilities. Nevertheless, each of the three modules that were covered in this study had many points of improvement. Starting with the road following module, one of the main tasks for improving precision is the implementation of odometry into the system. Having a consistent behavior of the motors will make the user take advantage of the trained model in this study. For the object/pedestrian recognition module there were two tasks pending to resolve. First, it would be beneficial to find solutions to improve pedestrian/object classification under backlit conditions. Secondly, it is crucial to implement image segmentation techniques to the model so it can detect multiple objects in one image. Finally, autonomous docking is still a task pending to develop. Even though without odometry this task was difficult to accomplish, the truth is that there is an opportunity of improvement in the exploration and optimization of fiducial markers utilization like Apriltags.

Final Thoughts

V2G is a technology that brings with it many potential benefits for both our citizens and the community. Some of them include being cheaper and faster than other energy storage devices, while also offering novel economic revenues to consumers and reducing the environmental damage from both the electricity and transport sectors. If we can understand and overcome the obstacles that this technology brings with it, not only the University of Texas at El Paso and its community could benefit highly from being research pioneers in the next energy revolution, but also any interested electrical and automotive companies will find technical, economic, and environmental solutions within their industry. For these reasons, creating a physical prototype that could simulate and test many scenarios around the implementation of V2G into our community will bring significant value and will serve as the first step towards the upcoming V2G technology implementation.
References


[34] Urquia, B. V. de. (2022, June 16). *Almost 400 automated vehicle crashes reported in the USA in the last year*. RSS. Retrieved July 25, 2022, from https://eandt.theiet.org/content/articles/2022/06/almost-400-automated-vehicles-crashes-were-reported-in-the-last-year/#:~:text=Automakers%20reported%20nearly%20400%20crashes,in%20the%20last%2011%20months.


Appendix A – Object Recognition/Pedestrian Reference Code

Training / Validation Code

NUM_EPOCHS = 100
BEST_MODEL_PATH = 'best_model_100.pth'
best_accuracy = 0.0

optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

handle = show(row(p1, p2), notebook_handle=True)

for epoch in range(NUM_EPOCHS):
    train_loss = 0.0
    train_error_count = 0.0
    for images, labels in iter(train_loader):
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = F.cross_entropy(outputs, labels)
        train_loss += loss
        train_error_count += float(torch.sum(torch.abs(labels - outputs.argmax(1))))
        loss.backward()
        optimizer.step()
    train_loss /= len(train_loader)

    test_loss = 0.0
    test_error_count = 0.0
    for images, labels in iter(test_loader):
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        loss = F.cross_entropy(outputs, labels)
        test_loss += loss
        test_error_count += float(torch.sum(torch.abs(labels - outputs.argmax(1))))
    test_loss /= len(test_loader)
    train_accuracy = 1.0 - float(train_error_count) / float(len(train_dataset))
    test_accuracy = 1.0 - float(test_error_count) / float(len(test_dataset))
    print('%d: %f, %f, %f, %f' % (epoch+1, train_loss, test_loss, train_accuracy, test_accuracy))

    if test_accuracy > best_accuracy:
        torch.save(model.state_dict(), BEST_MODEL_PATH)
        best_accuracy = test_accuracy

    new_data1 = {'epochs': [epoch+1],
                'trainlosses': [float(train_loss)],
                'testlosses': [float(test_loss)]}
    source1.stream(new_data1)

    new_data2 = {'epochs': [epoch+1],
                'train_accuracies': [float(train_accuracy)],
                'test_accuracies': [float(test_accuracy)]}
    source2.stream(new_data2)

    push_notebook(handle=handle)
Function to be called every time the value of the camera changes.

```python
1. import torch.nn.functional as F
2. import time
3.
4. def update(change):
5.     global blocked_slider, robot
6.
7.     x = change['new']
8.     x = preprocess(x)
9.     y = model(x)
10.
11.     # we apply the `softmax` function to normalize the output vector so it sums to 1 (which makes it a probability distribution)
12.     y = F.softmax(y, dim=1)
13.
14.     prob_park = float(y.flatten()[0])
15.     prob_free = float(y.flatten()[1])
16.     prob_pedestrian = float(y.flatten()[2])
17.     prob_other = float(y.flatten()[3])
18.     prob_other1 = float(y.flatten()[4])
19.     prob_other2 = float(y.flatten()[5])
20.     prob_other3 = float(y.flatten()[6])
21.
22.     print('%f %f %f %f %f %f %f' % (prob_park, prob_free, prob_pedestrian, prob_other, prob_other1, prob_other2, prob_other3))
23.     print('%f' % (prob_pedestrian))
24.
25.     time.sleep(0.0001)
26.
27.
28.     update({ 'new': camera.value })  # we call the function once to intialize
```
Appendix B – Road Following Reference Code

Training / Validation Code

```python
1. NUM_EPOCHS = 50
2. BEST_MODEL_PATH = 'best_steering_model_coordinates.pth'
3. best_loss = 1e9
4. optimizer = optim.Adam(model.parameters())
5. handle = show(row(p1), notebook_handle=True)
6. for epoch in range(NUM_EPOCHS):
7.   model.train()
8.   train_loss = 0.0
9.   for images, labels in iter(train_loader):
10.      images = images.to(device)
11.      labels = labels.to(device)
12.      optimizer.zero_grad()
13.      outputs = model(images)
14.      loss = F.mse_loss(outputs, labels)  # outputs(predicted values), labels(expected values)
15.      train_loss += float(loss)
16.      loss.backward()
17.      optimizer.step()
18.   model.eval()
19.   test_loss = 0.0
20.   val = 0.0
21.   act = 0.0
22.   for images, labels in iter(test_loader):
23.      images = images.to(device)
24.      labels = labels.to(device)
25.      outputs = model(images)
26.      predict = model(images).detach().float().cpu().numpy().flatten()
27.      actual = labels.detach().float().cpu().numpy().flatten()
28.      loss = F.mse_loss(outputs, labels)
29.      test_loss += float(loss)
30.      val = float(predict[0]+predict[1])
31.      #valy = float(predict[1])
32.      act = float(actual[0]+actual[1])
33.      train_loss /= len(train_loader)
34.      test_loss /= len(test_loader)
35.      print('%d: %f, %f' % (epoch+1, train_loss, test_loss))
36.      if test_loss < best_loss:
37.         torch.save(model.state_dict(), BEST_MODEL_PATH)
38.         best_loss = test_loss
39. new_data1 = {'epochs': [epoch+1],
40.             'trainlosses': [float(train_loss)],
41.             'verificationlosses': [float(test_loss)]}
42. source1.stream(new_data1)
```
Function to be called every time the value of the camera changes.

```python
1. angle = 0.0
2. angle_last = 0.0
3.
4. def execute(change):
5.     global angle, angle_last
6.     image = change['new']
7.     xy = model(preprocess(image)).detach().float().cpu().numpy().flatten()
8.     x = xy[0]
9.     y = xy[1]
10.
11.    x_slider.value = x
12.    y_slider.value = y
13.
14.    speed_slider.value = speed_gain_slider.value
15.
16.    angle = np.arctan2(x, y)
17.    pid = angle * steering_gain_slider.value + (angle - angle_last) * steering_dgain_slider.value
18.    angle_last = angle
19.
20.    steering_slider.value = pid + steering_bias_slider.value
21.
22.    robot.left_motor.value = (max(min(speed_slider.value + steering_slider.value, 1.0), -1.0))
23.    robot.right_motor.value = (max(min(speed_slider.value - steering_slider.value, 1.0), -1.0))
24.
25.    execute({'new': camera.value})
```
Appendix C – Apriltag Detection Reference Code

```python
1. parser = ArgumentParser(
2.     description='test apriltag Python bindings')
3. parser.add_argument('device_or_movie', metavar='INPUT', nargs='?',
4.     default=gst_str(),
5.     help='Movie to load or integer ID of camera device')
6. apriltag.add_arguments(parser)
7. options = parser.parse_args()
8. try:
9.     cap = cv2.VideoCapture(options.device_or_movie,cv2.CAP_GSTREAMER)
10. except ValueError:
11.     cap = cv2.VideoCapture(options.device_or_movie,cv2.CAP_GSTREAMER)
12. detector = apriltag.Detector(None,searchpath=apriltag._get_demo_searchpath())
13. while True:
14.     success, frame = cap.read()
15.     if not success:
16.         break
17.     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
18.     detections, dimg = detector.detect(gray, return_image=True)
19.     num_detections = len(detections)
20.     status_slider.value = num_detections
21.     print('Detected {} tags.\n'.format(num_detections), end='\r')
22.     for d in detections:
23.         (cX,cY) = (int(d.center[0]),int(d.center[1]))
24.         x_slider.value = cX
25.         y_slider.value = cY
```
## Appendix D – Bill of Materials

<table>
<thead>
<tr>
<th>Product</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveshare Jetbot Kit</td>
<td>Chassis, motors, camera, micro-SD, etc.</td>
<td>$120.00</td>
</tr>
<tr>
<td>Interlocking Exercise Foam Mats</td>
<td>18650 batteries x3</td>
<td>$26.00</td>
</tr>
<tr>
<td>Lithium Batteries</td>
<td></td>
<td>$35.00</td>
</tr>
<tr>
<td>Duct Tape</td>
<td>White, Red, Yellow</td>
<td>$30.00</td>
</tr>
<tr>
<td>Traffic signs play set</td>
<td></td>
<td>$17.00</td>
</tr>
<tr>
<td>Fisher-Price figurines (5)</td>
<td></td>
<td>$18.00</td>
</tr>
<tr>
<td>Lego figurines (20)</td>
<td></td>
<td>$25.00</td>
</tr>
<tr>
<td>Magnetic micro-USB adapter</td>
<td></td>
<td>$25.00</td>
</tr>
<tr>
<td>Adafruit NeoPixel Strip</td>
<td></td>
<td>$6.00</td>
</tr>
<tr>
<td>ESP32 Dev Kit</td>
<td></td>
<td>$10.00</td>
</tr>
</tbody>
</table>
Vita

Carlos Adolfo Cortes Pliego was born in Ciudad Juarez, Mexico where he completed his studies through high school. Thanks to his passion for science and technology he graduated top 5% from “Instituto Tecnologico y de Estudios Superiores de Monterrey” high school. Thanks to this, he was awarded the University of Texas at El Paso (UTEP) Presidential Excellence Scholarship. Mr. Cortes pursued a degree in Electrical Engineering at UTEP. During these period, Mr. Cortes was a member and vice-president of the University Soccer Team, member of the Institute of Electrical and Electronic Engineers (IEEE), and officer and president of the Society of Hispanic Professional Engineers (SHPE). Besides this, Mr. Cortes participated in multiple on-campus employments since 2017.

Mr. Cortes graduated from his bachelors at UTEP during the Spring 2020. Mr. Cortes graduated with Cum Laude honors. As an undergraduate student, Mr. Cortes participated in UTEP’s Fast Track program, starting with his master’s degree coursework, where he took neural networks and robotics courses and met who later became his mentors Dr. Patricia Nava and Dr. Robert Roberts. Mr. Cortes started his neural networks research activities with Dr. Nava in Fall 2020 where he became more knowledgeable on neural networks architectures and in Fall 2021, he extended his knowledge in robotics under the guidance of Dr. Roberts. During his master’s degree, he became Teacher Assistant for robotics class and maintained a 3.8 GPA. In 2021, Mr. Cortes participated in an internship as a Technical Program Manager with Microsoft. Mr. Cortes will join Microsoft in September of 2022 as a Technical Program Manager.

Contact email: crts.plgo@gmail.com