Vehicle Detection, Tracking, and Status Classification in Parking Lots

Project Proposal

Team name

Jackie Mioduski

Fabianna Barbarino

Spencer Cho

Aniruddha Srinivasan

Department of Computer Science

Texas A&M University

02/02/2023

Table of Contents

1 Executive summary 3

2 Introduction 4

2.1 Needs statement 4

2.2 Goal and objectives 4

2.3 Design constraints and feasibility 4

3 Literature and technical survey 5

4 Proposed work 7

4.1 Evaluation of alternative solutions 7

4.2 Design specifications 9

4.3 Approach for design validation 12

5 Engineering standards 13

5.1 Project management 13

5.2 Schedule of tasks, Pert and Gantt charts 14

5.3 Economic analysis 16

5.4 Societal, safety and environmental analysis 16

5.5 Itemized budget 17

6 References 17

7 Appendices 18

7.1 Bios and CVs 18

1 Executive summary

This project was assigned to us by a large physical retail organization with stores all across the United States. Many vehicles park there daily, and the parking lots are monitored by security cameras. Any incidents are caught using manual reviews, security guards. Parking layout improvements, analysis and design are all based on manual reviews. This process can be time and money consuming, and it can be difficult for both security guards and business analysts to get an understanding of incidents and patterns.

However, our customer believes that new technologies can make it radically easier for both security guards and business analysts to get a good understanding of their parking space and how exactly it's utilized. Instead of human review, they want us to train machine learning models to detect parked cars, illegally parked cars, stopped cars, and speeding cars. The live camera feed and associated data will then be sent to security guards, giving them the tools to quickly act. Business analysts will have a separate view where they can analyze peak traffic, unused parking spaces, etc. For scalability and cost purposes, our solution must use their existing setup of cameras with no additional sensors. Since infrastructure capabilities could vary by store, our application must be containerized and thus be flexibly able to run either in various clouds, or an on prem server.

In order to design this 'full stack' solution, we will need to train an ML model capable of quickly and accurately identifying cars and car status. We will also need to establish a storage and database schema that logs relevant information about the vehicle and parking space occupancy in a way that is easy to query. We will need two separate views, one for the security guard that is optimized for immediately actionable items, and another view for the business analyst that allows them to see long term trends. Finally, we will need to run this application inside a container, and establish the minimum hardware specs our solution needs to run optimally.

Our expected result at the end of this project is to be able to run this application with a single live feed analyzing 5 cars at a time as a minimum viable product. We expect that if implemented fully by our customer, they will see a huge increase in actionable intelligence and capability - with no additional hardware. This will allow security guards and parking officials to have faster response times and less manual review, potentially clearing up obstacles that allow stores to run smoother. Business analysts will have a wealth of data - potentially across multiple stores that will allow them to find patterns in parking space utilization and come up with ways to optimize the stores. Although there may be a cost for the infrastructure in order to run the application, the data that the application will provide will be a huge improvement to what the customer had previously.

Our team is well equipped to handle this project, which requires front end, backend and machine learning skill sets. We have programmers in each field, and we believe that we will be able to come up with a successful solution that will ultimately allow our customers to be more efficient and improve their services.

Introduction

Our main goal is to build a containerized app that has the ability to detect and track vehicles, log driving patterns, and monitor the parking lots with a graphic user interface (GUI). The camera should be able to track at least 5 vehicles at a time and the GUI should display the current status of vehicles. For example, a vehicle could be parked, illegally parked, stopped, driving, speeding, etc. This would allow us to solve problems like traffic, illegal parking, and crashes in real time. We plan on working with existing vehicle detection programs like OpenCV. This means our backend will be implemented using Python. We will also keep track of the cars using bounding boxes. This information will be fed into a database which will allow us to later visualize historical patterns for specific cameras. The GUI will be implemented using REACT, HTML5, and CSS.

1.1 Needs statement

According to the National Safety Council (NSC), “tens of thousands of crashes occur in parking lots and garage structures annually, resulting in hundreds of deaths and thousands of injuries” [1]. While not all of these can be prevented, the ability to monitor and recognize traffic patterns in parking lots can greatly help avoid these kinds of accidents from occurring. Moreover, the security personnel watching over the parking lot could have a quicker reaction time if notified immediately when something occurs. On top of this, traffic has been an ongoing aggravation for many drivers and stores. It keeps people from getting to the store smoothly and having a pleasant shopping experience. Monitoring the traffic patterns in a parking lot can allow us to find ways to improve the flow of traffic and ensure the customers are satisfied.

1.2 Goal and objectives

Our goal is to build a containerized app that detects and tracks vehicles across a parking lot and continually logs the driving patterns. This can all be monitored using our user interface which allows two different types of users to view it: security personnel, and data scientists. The security personnel use the GUI as a way to monitor parking lot activity and ensure the safety of all the customers. The data scientist uses the GUI to mine the database and analyze patterns like where the traffic jams are occurring and why. Our first objective is to ensure our vehicle detector can track at least 5 cars. The cameras will be looking out at around 12 feet off the ground, and we must ensure they can successfully track at least 5 cars at this height. Our second objective is to have bounding boxes around the vehicles whose coordinates will be sent to the database. Our third objective is to tie the backend with the front end successfully for our GUI. We want to ensure that the GUI works in real time, so that the security personnel have the correct information in a timely manner. Our fourth objective is to ensure our implementation works as a desktop app on a GPU workstation or laptop.

1.3 Design constraints and feasibility

Some of our technical constraints are using a USB camera and the minimal testing videos that we will be provided. This will not be a big issue for our objectives because the camera works well enough for our needs and we plan on creating or finding more testing videos. The main physical

constraint is the camera being 12 feet high and looking out. This is something we will have to work with so our objectives can be achieved, but it will not be something that stops our progress. Our economical constraint is that we have a limited budget per person on our team for all of the expenses we will have for the project. We are building a budget and plan to leave room for emergency expenses, so this will most likely not affect our completion of our objectives. Last but not least, our temporal constraint is that we have until April to work on the project. This will most likely be our most impactful constraint and will hurt our ability to complete our objectives if we fall behind.

2 Literature and technical survey

There are several commercial products and research projects already in place to solve the issue of parking lot monitoring and optimization for businesses. A few are detailed below.

- **M-Gage Node Pucks**:
  M-Gage Node pucks are sensors used to detect a vehicle in the vicinity via fluctuations in Earth's magnetic fields caused by any "large ferrous objects." They can be placed under the ground in parking spaces to provide a system to direct vehicles to available parking. This is a costly solution to reducing parking congestion and customer frustration which requires drilling a small hole in the concrete of each parking spot. These sensors would run upwards of $436 per parking space, not to mention costs associated with drilling into concrete: 3-inch core equipment, sealing material, etc. [2].
  This solution also provides limited data about traffic flow/safety/tracking within the parking lot unless you install multiple sensors in the lanes as well; and does not provide data on the individual status of vehicles so much as the presence of vehicles in specific locations. Therefore this is an overall poor solution as it is not cost effective, easily implemented, or scalable and it has limited applications to our problem needs.

- **Parksol**: Parksol is a fully managed product that offers integrated analytics that give more information as to parking habits, APIs, and sells both the sensor hardware and the software. It's designed for our customer, but perhaps with not as much scale. As a fully managed product, our customer will not have to spend development time to maintain and improve the product. Parksol also runs on-prem which means data is kept secure [3].
  Integrated analytics will help satisfy the business analyst's needs. However, we may have features in this product that we don't need and the product is mostly for smaller scale businesses and has not proven its ability to meet our customer's needs. Since our business has many stores country-wide, a custom solution will achieve meaningful cost savings.

- **ArcVision, A Deep Learning based Illegal Parking Detection Platform**:
  A 2019 Esri Intern Hackathon Project defines a camera-based solution to track illegal parking offenders while recording license plate numbers for vehicle identification. This solution uses OpenCV for video image extraction, Darknet YOLOv2 for vehicle extraction, Keras and Tensorflow for license plate detection, and ALPR (Automated License Plate Readers) for detecting license plate numbers. The user interface includes a Map View (for spatial data

visualization and user-defined illegal zones for parking) and a Card View for video feed and parking classification statistics [4].

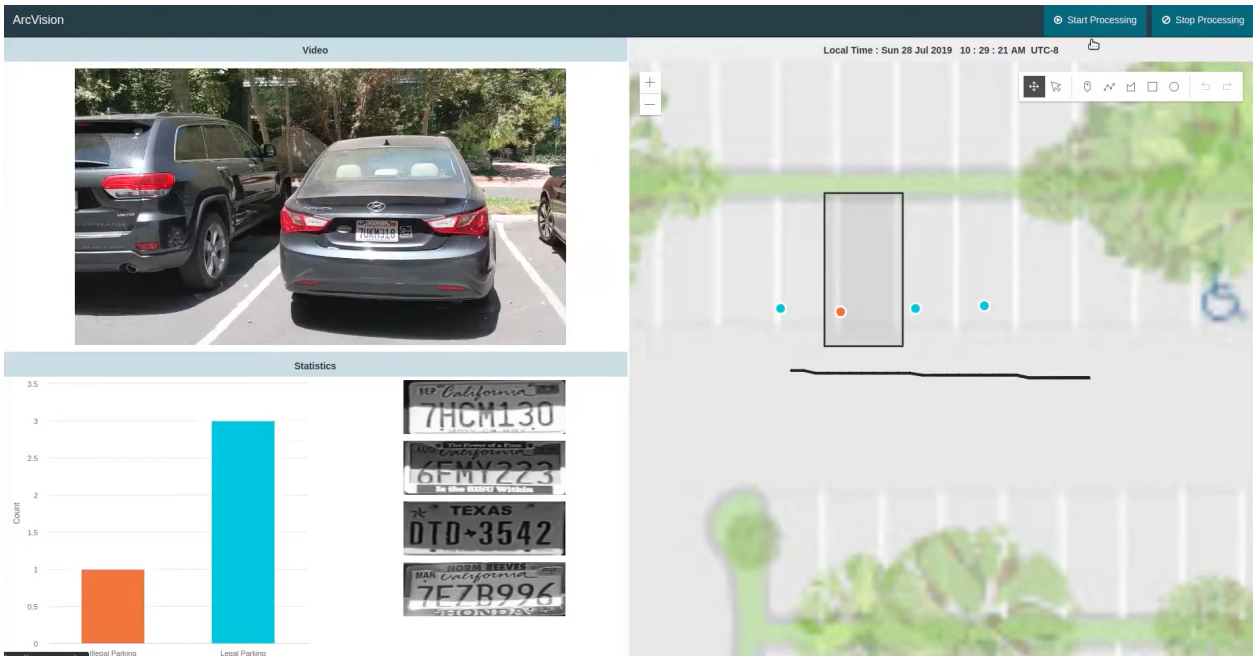


Figure 1: ArcVision UI
Source: Adapted from [5]

- **Smartpark System - Smart Parking**:
  Smart Parking provides many parking monitoring services and devices across the UK from vehicle sensors to mobile apps to Cloud interfaces for accessing and processing data in real time. The Smartpark system is fully integrated with parking guidance, payment, and analytics with the option for hiring human patrols to manage parking [6].

- **An Ultrasonic Sensor System for Vehicle Detection Application**:
  A team of researchers from the Department of Physics in the Bandung Institute of Technology developed an ultrasonic sensor system to count and classify passing vehicles by type (Sedan, SUV, or Truck) and speed. Their system used two ultrasonic sensors, an Arduino microcontroller with data collection capabilities, and a Java program. Their solution does not rely on video quality as a camera would and it is smaller and cheaper as well. Although unlike a camera, it only functions to monitor vehicles passing a certain point [7].

These designs relate to ours in that they attempt to solve similar issues with vehicle detection and parking lot management, albeit using different technology. Our proposed design is most similar to the ArcVision camera-based solution above and it should be cheaper than the sensor-based models, and more applicable to the needs of security as well as. Our solution will be better for tracking and

classifying the status of multiple vehicles at once over a wide area and gathering data about traffic flow/congestion over time as well as logging illegal activities.

Proposed work

2.1 Evaluation of alternative solutions

Any solution that we come up with will have many components such as a detection system, a tracking system, a backend to store information, and a Graphical User Interface for both security guards and business analysts. The following alternative solutions will focus on a few different specific detection and tracking algorithms that we may choose from and their pros and cons.

Detection Algorithms:

YOLO (specifically YOLOv7) -

> **Pros**: YOLO is the best open-source neural network model for object detection that is capable of drawing bounding boxes over vehicles in real time. It is designed for speed and low infrastructure use so it is perfect for optimizing cost. We will also easily be able to send data to any backend we decide to choose.
>
> **Cons**: The algorithm is more sensitive to noise and not as accurate when estimating object area. It also struggles when detecting small things in images. The YOLO detection model is frequently updated. Retraining the model requires at least some level of software engineering expertise. The company for which we are building our solution has to manage their own infrastructure, maintenance, and updates.

Mask R-CNN -

> **Pros:** Masking allows for a better approximation of area when compared to YOLO.
>
> **Cons:** It is not as fast or accurate for detecting multiple objects in real time, so it is not the best option for our purposes.

Conclusion -

Weighing the pros and cons of both of these options, we are leaning more toward choosing YOLO over Mask R-CNN because YOLO will better suit our needs for efficient real-time vehicle detection.

Tracking Algorithms:

DeepSORT -

> **Pros:** DeepSORT has respectable tracking accuracy. It is also a well known algorithm as an improvement to SORT. There may be more online resources and tutorials to help apply this algorithm to our solution because it has been around longer.
>
> **Cons:** DeepSORT suffers a low FPS when compared to the older SORT and newer tracking algorithms such as ByteTrack and SMILETrack when tested on the MOT17 benchmark [8]. It

may still be able to surpass our benchmark of greater than 10 FPS per second with 5 vehicles in frame but this solution is not very scalable.

ByteTrack -

Pros: It has higher accuracy performance metrics than DeepSORT as well as a higher achievable FPS. It is relatively new, but there is enough documentation and tutorials from multiple sources to support learning this method.

Cons: It is accurate but it is not the most accurate solution available to us. Cutting-edge algorithms such as SMILEtrack can outperform this now.

SMILETrack -

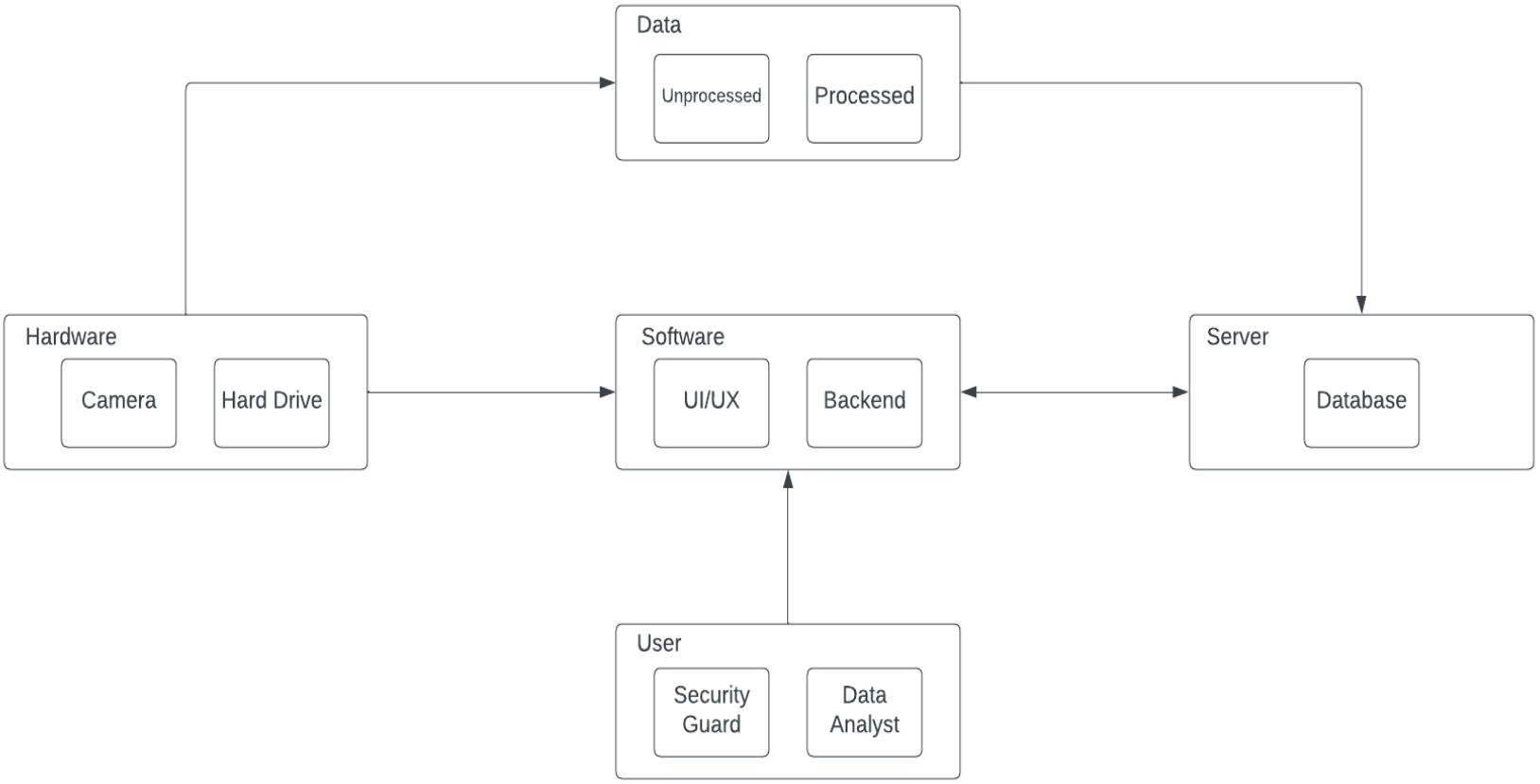
Pros: SMILETrack ranks the highest on the MOT17 benchmark [8]. It is adapted from YOLOv7 and ByteTrack so we may be able to gain some more insight into the inner workings of the code from looking at those other models.

Cons: Further testing is required to figure out whether the FPS achieves our standards when detecting multiple vehicles. It will have the fewest assistive resources because it is so new.

Conclusion -

All of these tracking algorithms are open source and freely available on Github. We would like to try to implement our solution with the fastest, most cost-effective, scalable algorithm so we are leaning towards SMILETrack. If this algorithm proves to be not as optimized as we need it to be in terms of maintaining FPS, then we may use ByteTrack instead.

2.2 Design specifications



High Block Diagram Descriptions:

Hardware Module:

For the Hardware portion of our Block Diagram, this would consist of our physical media utilized for the project. For example, any cameras, hard drives, computers, etcetera… These would fall under this module. The parts of this module currently would include a camera and hard drive, and possibly more if we see the need for anything else throughout the development of the project. The camera would be used as our eyes, in order to gain information over a parking lot. How cars are parked, if they are parked illegally, if they are moving, if there is a traffic jam, so on and so forth. Next, is our hard drive, this is fairly simple and will be a method of storing our data if needed, as well as using our application on other devices in order to test the functionality of how our application may run given different devices. The information garnered from the camera would be sent as Data to our Server and would be used as we see fit.

Data Module:

The Data module is very simple, but without it we would not be able to continue throughout our project whatsoever. To start off, the data gathered from the cameras of our Hardware portion of our Block Diagram is taken, purely as unprocessed, unfiltered data, and would then be sent to our Server in order to house this information until we decide to use it and determine what is occurring in the data itself, which would take place in our application via machine learning techniques. With this, while it may seem like not much, would be a major source in how we may be able to create our own model to use for our machine learning as we progress through the development process of our project.

User Module:

The User module is how a User will interact with our application. And how our entire project is made useful, is all dependent on the User. The User will mainly consist of two positions or viewpoints as of the beginning of our development, a Bodyguard view and an Analyst view. The idea is that the Bodyguard's role as a User is to use the information gathered from the Hardware and Software portion of the project in order to take note of any mishaps that may occur while they are on the job, which would most likely occur from the viewpoint of live video feeds. For example, if our application determines that a vehicle is illegally parked, the Bodyguard would in some way be notified so that they may be able to take the appropriate action required to remedy the situation. The same can be said if there was an accident, and so on. This notification while not currently set in stone can be a sound alert, a text message, an on screen display notification, etc… On the other hand for an Analyst the use of this application completely changes. Their goal will be to analyze the data from the Data and Server module, and take note of any unusual occurrences that our application may find, therefore their viewpoint would revolve more around the Data and less on video feeds.. For example, there may be an influx of cars causing a traffic jam at a certain spot, the Analyst would use our Data and determine why that may be occurring and try to solve it, or there may be a location prone to numerous accidents, and again the Analyst would be in charge of trying to fix this issues in some way or form. The Data the Analyst uses would be gathered and organized in some way to make this process easier via algorithms we end up implementing.

Server Module:

The Server module consists of a database that would house not only the untrained data from the video feed, but the processed data used to train our application. The software would pull the untrained data from the server side, process it using machine learning algorithms, and push the processed data back into the database. Live video feed would be taken and transformed into data that would allow for further training of the model which is then stored in the server. The server would provide a consistent benchmark for the software by allowing us to view the training unfold over time.

Software Module:

The Software module consists of both a frontend and a backend. The backend will do much of the processing of data utilizing the programming language Python. Within Python, various machine learning modules will be used such as pytorch, sklearn, and pandas in order to train our application to properly identify vehicle positioning. This processed data will in turn be sent to the server for storage and there is hope that following training using recorded video, we will be able to transition to a live video feed for real time data processing. The software will primarily be focused on that of machine learning, a subset of artificial intelligence, that allows for the training of an application to classify and predict scenarios based on predetermined models. This involves using features and labels in order to fit data to models we deem appropriate and should the application be incorrect in its determination, we shall alter the program and the model to fit our specifications. We will also use Docker in order to containerize our application, making the testing, building, and deployment easier all the while allowing for version control. In addition to a backend, a frontend consisting of custom components created within React.js will also be created in order to make navigation and use of the application easier on users.

2.3 Approach for design validation

If we divide our project into Tracking, Database, GUI, Infrastructure - here are the tentative metrics for success.

Tracking:

Camera should be able to track at least 5 vehicles at an FPS greater than 10. Should use a contemporary ML model. Should detect parked, stopped, not stopped cars. The accuracy should be reasonably good (85%). According to our project manager, this is a priority for the application.

Database:

Log driving patterns with x,y and time data. The database should also make sure that identifying information is not present. Temporary ID from Deepsort is ok and does not need to persist. Queries should be able to extract historical patterns for specific cameras. Currently the plan is to use a SQL database, but we may use MongoDB

GUI:

There should be two views: one for a security guard, and one for a business analyst. The security guard view should be tailored towards live, ondemand footage. Security guards would be able to view a live feed. They should also be able to easily see incidents sorted by time. Clicking on an event should bring up the video recording at the time. The business analyst view should focus more on historical data as well as patterns. At minimum they should see a table where they can sort and filter by time/event.

In terms of technology, a web app is preferred, but is a stretch goal.

Infrastructure/Deployment:

Our app should be able to run in a Docker container. The intent of this is so that our project assigner can run our application on any infrastructure/cloud environment they deem necessary. Additionally, we should find minimum specs needed so that our application will run smoothly, and explore the relationship between increasing specs and the speed of our ML detection. It is likely that we will need infrastructure with a dedicated GPU for best performance in ML.

Testing:

Once we have established our application, we want to be able to test and see if it conforms to our project expectations. We will be using a single usb webcam to simulate a live feed, mounted at a similar location to how our project assigner will mount them, over a parking lot. Alternatively our project manager cleared us to use an mp4 file that is fed to our application as if it were live.

We will then use our application to view this live feed, and track the accuracy and speed (at least 10FPS for 5 cars) . Security guards should be able to view any incidents by clicking on a particular incident and watching a replay. We will also use the Business Analyst view in order to track historical data, in this way testing the database. Our app should be running in a container.

After discussion with our project manager, the strictly necessary events to track are **stopped/parked/not stopped**. Speeding and illegally parked are out of MVP scope as they

may require more advanced analysis. However, if the application is fully running with the necessary events we may add them later on

If our application is able to successfully complete this test, we will consider that to be our MVP (minimum viable product). There are other potential considerations when looking at a customer that operates on the scale of ours. but we consider this a good benchmark for a proof of concept.

Summary:

App must run in a container

App must detect stopped, not stopped, parked

App must have a database with x,y,time,event

App must have two views

3 Engineering standards

3.1 Project management

Fabianna Barbarino worked as a Software Engineering Intern at Publicis Sapient during the Summer of 2022. During this employment, she worked on an Agile team to provide a fully responsive web application to their client, Chevron. Her experience of developing responsive websites is further strengthened by two personal websites and she has experience working with databases from a sprint planner course project.

Spencer Cho has experience working with front and back end development through Texas A&M coursework. He also interned at Findhelp during Summer 2022, at which he helped create a tool to ease the workflow between two teams.

Jacqueline Mioduski is interning at Atos and is building servers, managing approvals, decommissions, and monthly summary reports. This is supplemented by experience working on front end web application design and development.

Tyler Roosth has experience working on back end development over various projects, with personal projects focusing on front. He has experience assisting with user-application interactions via employment with Texas A&M's IT Department.

Aniruddha Srinivasan has completed 5 internships with various companies in the Computer Science field. During this time, he has gained experience in working with cloud and software development.

Coleman Todd has experience working in front and back end development over multiple projects and with different development styles. Through personal projects, he has worked on the storing and use of user inputted data for analysis by a managerial user.

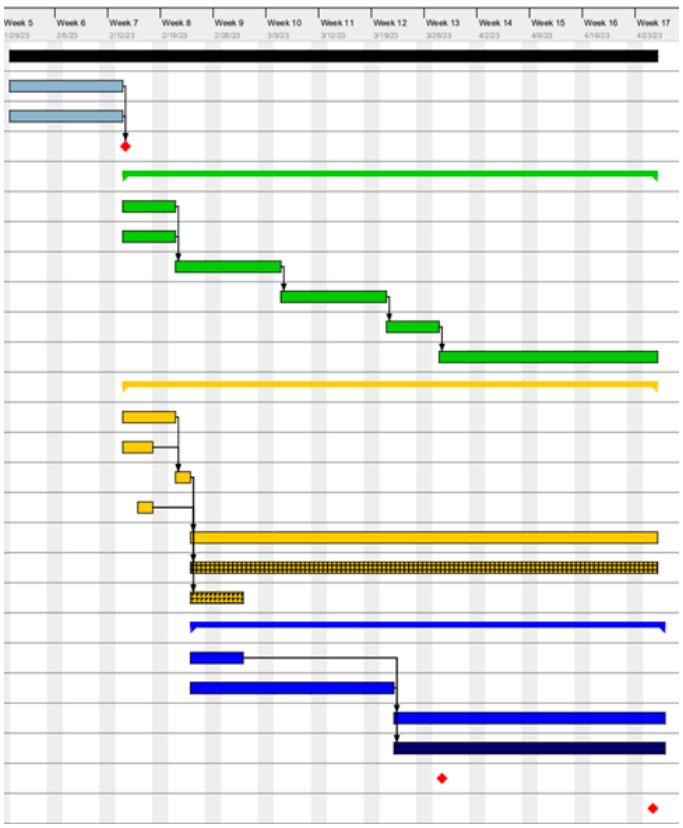
While there will be division of primary responsibility, the team will try to maintain a fluid status of which module they will be working on during a given week. These divisions are to establish which members of the team are to be in charge of overall structure for consistency within the module. For the machine learning and data processing section of the software module, Spencer Cho and Tyler Roosth will be the two charged with heading that portion of the project. Data storage, organization, and access will be headed by Aniruddha Srinivasan and Coleman Todd. Finally, the front end and user interface implementation will be led by Fabianna Barbarino and Jaqueline Mioduski. These roles are guidelines and during weeks where the tasks scheduled do not include a person's given responsibility, then they will be tasked to work under a different section as needed. Weekly reporting will be cycled through the different team members, with notes and input from other members supplementing these reports as needed.

Project management is structured around having 3 meetings throughout the week, weekly written reports, and maintaining clear communication between task workers. The first meeting of the week will be with our supervisor and client so that we can update them on project progress and clarify any directional changes that might need to be implemented. Our two other meetings will be near the middle and end of the week, and their purposes are to do internal progress analysis and goal setting for the next week respectively. These meetings will be conducted as an open forum so that any problems or adjustments to tasks can be discussed with everyone present and knowledgeable about the situation. Weekly reports are done so that we have a written record of goals set and changes made, that way it can be easily referenced at a later date. Finally, clear and consistent communication will allow for any problems or needs present in one section of the project to be known by all members of the team. This should minimize instances of incompatibility between modules being implemented.

3.2 Schedule of tasks, Pert and Gantt charts

Due to the larger scope of our project, we have divided our project into three simultaneous and codependent tasks. Each of these subtasks can be treated like their own separate Gantt charts, with each involving the development and implementation of different modules. The Gantt chart is presented below:

Name	Begin date	End date
Vehicle Classification Project	1/30/23	4/25/23
Project Proposal	1/30/23	2/13/23
Project Proposal Slides	1/30/23	2/13/23
Project Proposal Presentation	2/14/23	2/14/23
Vehicle Classification Machine Learning	2/14/23	4/25/23
Organize Machine Learning Data	2/14/23	2/20/23
ML Model Brainstorming	2/14/23	2/20/23
ML Model Implementation	2/21/23	3/6/23
ML Model Training	3/7/23	3/20/23
ML Evaluation	3/21/23	3/27/23
ML Tuning	3/28/23	4/25/23
Database	2/14/23	4/25/23
Database Schema Design	2/14/23	2/20/23
Database Server Setup	2/14/23	2/17/23
Database Table Creation	2/21/23	2/22/23
Database I/O Format	2/16/23	2/17/23
Database Management Software I...	2/23/23	4/25/23
Security User Data Access Organization	2/23/23	4/25/23
Analyst User Data Access Organization	2/23/23	3/1/23
UI	2/23/23	4/26/23
User Interface Sketch	2/23/23	3/1/23
User Interface Prototype	2/23/23	3/21/23
User Interface Security	3/22/23	4/26/23
User Interface Analyst	3/22/23	4/26/23
CDR Presentation	3/28/23	3/28/23
Project Demo	4/25/23	4/25/23



The Machine Learning tasks are the most important set of tasks, as they make up the Vehicle Identification part of the Backend Module. It begins with the team deciding on what we want to have the architecture recognize and record. Then we will be organizing the data that we are going to be using as the training data for the recognition software. Next will be the implementation task followed by cyclical evaluation and tuning of the results given. This cycle is continued throughout the time frame we will be working on the project, with each cycle having a goal of better recognition or adding in a new behavior that it can categorize.

The Database tasks' importance is ensuring that all information, both raw and processed, will have a well maintained and easily workable environment. We begin by designing the schema we will be using for the project, then setting up the server and tables. We then want to design and format how the backend will be communicating, inserting, or updating data in the database before implementing the access protocols for our two views. These tasks will remain for the remainder of the project, as they are to be developed alongside our machine learning pipeline and information given or needed might change as that is developed.

Finally, our UI tasks are a catchall for our front-end software implementation. At the beginning of the project's life cycle, we will only be trying to get rudimentary UI made so that we can represent the data and structures we will be making in the backend. However, once the backend software is closer to our end goal criteria, we will begin to implement our two different views into the application.

As stated earlier, developing and training our ML pipeline is the most critical set of tasks as that is the basis by which we are able to accomplish the other subtasks. Division of tasks will be loosely

associated with the designated module leaders established in 5.1, however some flexibility is likely to be seen.

3.3 Economic analysis

If our product were to go commercial, we would be tapping into the security camera market which had the size of 6.40 Billion USD in 2021 [9]. Since we will be using third party USB cameras, we will have to get a manufacturer in order to reduce the cost of camera acquisition and thus reduce the overall volume production costs. The USB cameras required for our product are very generic and can be found from multiple vendors. This gives us a lot of flexibility in being able to find the best prices and best camera quality. The other parts of are system are software like a database hosted on the cloud which are also available from more than one vendor (Amazon, Google, Azure, etc). In order to ensure our product is working well we will need to have certain maintenance strategies set up. First of all, we will need an IT team that can be available for support of the software side of the product on a case by case basis. This can include situations where the cameras stop working all of a sudden or a certain part of the program is no longer responding. Secondly, we will need to ensure we have a service set up where customers can report cameras which have stopped working all together and need to be replaced. Something important to consider is any compliance to regulations that we must follow. There are no federal laws that prohibit video recording in public. Individual states have laws that state video taping is illegal if the individuals have a reasonable expectation of privacy which does not apply to parking lots. On top of this we do have to consider FCC regulations on surveillance cameras. On November 25, 2022 the FCC announced that "... some future equipment from Huawei, ZTE, Hytera, Hikvision and Dahua won't be authorized for sale in the US" [10]. This means we must ensure none of our equipment comes from these companies.

3.4 Societal, safety and environmental analysis

The potential beneficial impacts to society from our project is an increase in parking lot safety and a more efficient shopping experience. As previously mentioned, the monitoring of parking lots will allow for a quicker response time from security personnel in the case of an accident or illegal parking. This ensures the safety of customers is protected. Moreover, the monitoring also allows for improvement of traffic patterns which allows for a more satisfying and efficient shopping experience overall for individuals. The detrimental impacts to society include a potential loss of privacy. We will be recording the parking lot at all times which means there will be information stored on the cars and people will be on camera. When working on this project we must take some safety precautions. Firstly, we must be careful when conducting our own tests and setting the camera up 12 feet high. We have to ensure we are doing this safely with a ladder and there is a team member holding the ladder at all times while setting up. Secondly, since the cameras will be outdoors there is some possible damage to the camera due to inclement weather. We will have to ensure that the equipment is waterproof and windproof. Last but not least, we have to consider our environmental impact. According to an article published by Purdue University, "Just one hour of videoconferencing or streaming, for example, emits 150-1,000 grams of carbon dioxide (a gallon of gasoline burned from a car emits about 8,887 grams), requires 2-12 liters of water and demands a land area adding up to about the size of an iPad Mini" [11]. This means that our solution would generate an incredible amount of carbon emissions. A possible way to minimize the amount of emissions we produce would

be to only have the cameras turned on during business hours. There is a greater chance of the parking lot being empty outside of these hours, and even if there are cars there would be minimal activity which is less pertinent to our problem.

3.5 Itemized budget

- USB Camera - 50$
- External Hard drive - $100
- Potential computing costs (Linux workstation with GPU) $100

4 References

(https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf)

[1] "Parking lots & distracted driving," *National Safety Council*. [Online]. Available: https://www.nsc.org/road/safety-topics/distracted-driving/parking-lot-safety. [Accessed: 02-Feb-2023].

[2] "Wireless M-Gage Node," *Banner Engineering*. [Online]. Available: https://www.bannerengineering.com/us/en/products/wireless-sensor-networks/wireless-sensors/wireless-m-gage-node.html?sort=1#all. [Accessed: 12-Feb-2023].

[3] "Monitoring software," *Parksol*, 26-Aug-2022. [Online]. Available: https://parksol.lt/solutions/monitoring-software//. [Accessed: 12-Feb-2023].

[4] Z. Yin, H. Xiong, X. Zhou, D. W. Goldberg, D. Bennett, and C. Zhang, "A deep learning based illegal parking detection platform," *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, 2019.

[5] *A Deep Learning Based Illegal Parking Detection Platform*. YouTube, 2019.

[6] S. parking, "Smartpark system," *Smart parking*, 07-Apr-2022. [Online]. Available: https://www.smartparking.com/smartpark-system. [Accessed: 12-Feb-2023].

[7] R. Stiawan, A. Kusumadjati, N. S. Aminah, M. Djamal, and S. Viridi, "An ultrasonic sensor system for vehicle detection application," *Journal of Physics: Conference Series*, vol. 1204, p. 012017, 2019.

[8] "Papers with code - mot17 benchmark (multi-object tracking)," *The latest in Machine Learning*. [Online]. Available: https://paperswithcode.com/sota/multi-object-tracking-on-mot17. [Accessed: 13-Feb-2023].

[9] Emergen Research, "Security cameras market size, share: Industry forecast by 2030," *Security Cameras Market Size, Share | Industry Forecast by 2030*. [Online]. Available: https://www.emergenresearch.com/industry-report/security-cameras-market#:~:text=Market%20Synopsis,18.7%25%20during%20the%20forecast%20period. [Accessed: 11-Feb-2023].

[10] D. Hardawar, “FCC Bans Telecom and video surveillance gear from Huawei, ZTE and other Chinese companies,” *Engadget*, 25-Nov-2022. [Online]. Available: https://www.engadget.com/fcc-officially-bans-telecom-and-video-surveillance-gear-from-several-chinese-companies-003040729.html. [Accessed: 11-Feb-2023].

[11] P. N. Service, “Turn off that camera during virtual meetings, environmental study says,” *Purdue University News*. [Online]. Available: https://www.purdue.edu/newsroom/releases/2021/Q1/turn-off-that-camera-during-virtual-meetings,-environmental-study-says.html. [Accessed: 02-Feb-2023].

5 Appendices

5.1 Bios and CVs

Fabianna Barbarino:

Fabianna Barbarino is currently a senior majoring in Computer Science. This past summer she interned as a Software Engineering Intern at Publicis Sapient which is a technology consulting company. There, she worked on an Agile team alongside other engineers to provide Chevron with a fully responsive web application. She has also programmed two responsive personal websites, worked on a team to build a point automation sheet system for SHPE, and built a sprint planner for a databases course. She is also a National Hispanic Scholar.

Resume Link - LinkedIn

Jackie Mioduski:

Jackie Mioduski is a senior achieving a Bachelor of Arts in Computing at Texas A&M. She has worked on many computer science projects, specializing in front end web/application design and development. Currently, she is interning at Atos, building agency servers and managing their approvals, decommissions, and monthly summary reports.

Resume

Aniruddha Srinivasan:

Ani is a senior studying Computer Science. He has completed multiple internships in the Computer Science field. He has experience in cloud, software development.

Tyler Roosth

Tyler is currently a Senior pursuing a Bachelor of Science in Computer Science. He's worked on a few projects focusing more towards the back-end of development as well as some side projects that lent more towards the front-end. He works with the Texas A&M IT Department, focusing on helping students and professors with issues concerning their software.

Spencer Cho

Spencer Cho is currently a Senior at Texas A&M University pursuing a Bachelor's of Science in Computer Science. He has worked on multiple projects at Texas A&M University on both the frontend and backend side of development. In addition, this past summer, Spencer interned at Findhelp and helped create an internal tool used to ease workflow for two teams.

Coleman Todd

Coleman Todd is currently a Senior at Texas A&M University pursuing a Bachelor's of Science in Computer Science. He has worked with both frontend and backend development throughout multiple projects while at Texas A&M. He has also worked on personal projects that focus on storing user inputted data for managerial analysis of trends and performance.