Parking Availability System

Design Prototype Report

Erik Meurrens, Benjamin Simonson, Ryan Jalloul, Evan Tobon, Samer Khatib

**Introduction**

**Theoretical Background**

Currently, there is some trouble in running the YoloV11 LPR object detection model on the Raspberry Pi Model 3 B+ (RPi). This is crucial to the overall success of this project as running the model on the RPi will serve as our “eyes and ears” for the project. The model is currently working, as seen in figure 1, however, there is some trouble loading it and installing it on the RPi. Our group is confident that this can be done given that many others have gotten similar models to work on the same model as our RPi. You can reference the link [here,](https://docs.ultralytics.com/guides/raspberry-pi/) to refer to the same tutorial that our group has initially followed to configure the RPi for usage with the YoloV11 model. Currently, the model is being attempted to run on a docker container that contains the necessary libraries to run the model, however, the group is currently experiencing crashing when trying to just import the library required to make inferences with the mode as seen in figure 2. For further proof, you can refer to this [video,](https://www.youtube.com/watch?v=SILmGmR6S_8) which proves that the mode can in fact be run on the RPi.

Figure 1. Car object detection



Figure 2. Failed/Freezing model load

**Domain & Prior Art**

Currently, there are existing implementations that provide onsite observability for garage capacities. The solution currently employed by the University of Florida is the Genetec AutoVu system that uses an automatic license plate recognition service to identify and read license plates and count vehicles that are currently in the facility. While effective in license plate identification, this solution lacks a virtual view of parking facility capacities and current availability. This omission inconveniences drivers searching for available parking when none exists. Additionally, the current system comes with a significant cost burden and is facilitated via subscription services that may go underutilized. Specifically, this system maintains a “$6000 servicing cost, $11,000-$8,000 per lane, and $2,495 per camera” [1]. In contrast, our project will provide a much more affordable solution to the AutoVu as it will only require equipment and installation fees. Our project will also provide virtual observability into parking facility current availability to streamline an individual's search into finding a parking spot.

**Empirical Evidence**

**Architectural Elements**

**External Interface**

**Usability**

The external interface relies on a sensor module and camera for detecting and recording vehicles entering and exiting the parking facility. When a vehicle disrupts the sensor beam, the sensor triggers the camera module to begin capturing a video feed of whatever entity disrupted it. This video feed instantiates a license plate object detection model and will observe the frames provided by the video feed from the camera. Our group makes the assumption that all motor vehicles will have a license plate, if an entity does not have a license plate, then it is not a motor vehicle. If the model successfully interprets the provided frame and identifies a license plate, the features of the license plate, such as the text, are extracted. A PostgreSQL database is updated accordingly with the interpreted/classified license plate value. It is also updated with the respective entity either leaving or exiting the facility depending on which RPi has detected and classified the entity.

To communicate parking availability to users, a mobile application interface has been developed using the Flutter and Dart framework. The app reads directly from the PostgreSQL database, providing a real-time display of parking space availability for users on their devices.

Network connectivity is established via the Raspberry Pi’s onboard wireless module (model 3 B+). So far, there have been no issues with network connectivity from the RPi at any of the garages tested. More importantly, the garage we intend to roll out the solution to, the Reitz Union garage, provided more than sufficient network connection and served as the main supplier for the data/images that the model was trained on. This connectivity supports data transfer to the PostgreSQL database that is being hosted via AWS.

**Persistent state**

The persistent state(s) of the model will be to remain in a “sleep” mode until the sensor on the RPi is disrupted. A persistently running loop will poll for the sensor value and if it receives the appropriate value, the RPi camera will be turned on and the video feed will begin along with the model interpreting the video feed. Once the RPi is done capturing the feed, the images are interpreted, and the PostgreSQL

**Internal Interface**

**Components**

**Communication Mechanisms**