“Park-It!”: Midterm Project Report

Elijah Olsson, Subhashree Susindran, Jake Wantz

"Park-It!" GitHub Repository

The Project

- Use computer vision to easily check parking status and display data to website.
- Objectives:

1. **Build application software**: Space occupancy and object detection components.
2. **Send data**: Create data stream to remote web server (Pluto).
3. **Display data to end-user**: Develop webpage for viewing live and graphic data.

“Park-It!” over others

- Low-complexity, less dependencies
  - Simple setup
  - Model training not required
- User-friendly
  - Easy to generate and access data

https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

Current Project Plan

Backend

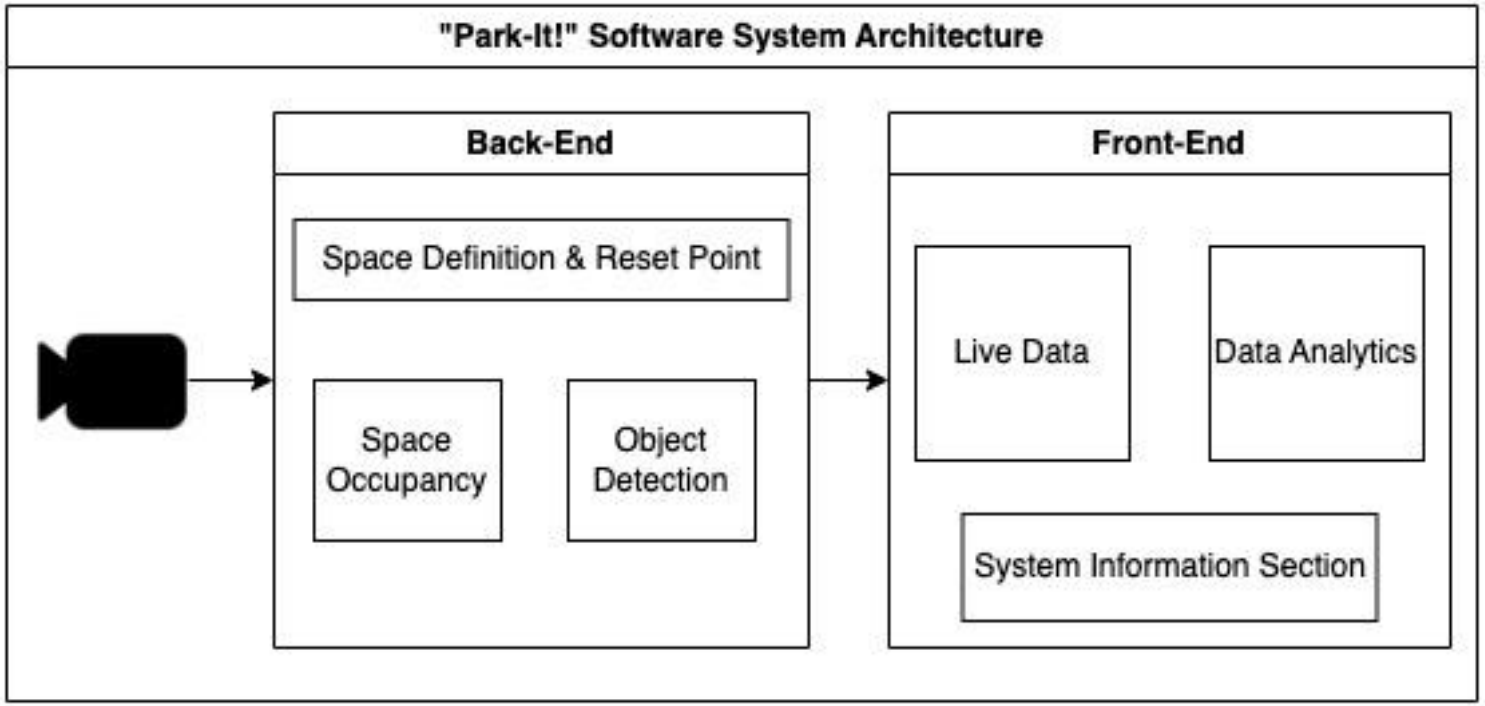
- Space Occupancy and Object Detection System
  - OpenCV (Computer Vision) library
  - YOLOv5 (You Only Look Once) model
- One-way communication tunnel to Pluto

Frontend

- Website on web server
  - Pluto for convenience & reliability
- Live data view
  - Live parking space status
- Data analytics view
  - System status & time charts
- System weather

Technical Description

- Monitor single parking space using OpenCV and YOLOv5 Python libraries.
- Occupancy and object detection for system validation.
- Transmit real-time system data to Pluto for accurate updates.
- Display parking space availability for online public access.
- Matplotlib for concise graphical reports on parking space status.



System Architecture Diagram (simplified)

Accomplished Work - Backend & Source Code

- 230 lines of code
- No database tools needed, textual data sent to Pluto directly.
- System Parameters:
  - Monitor video frame only inside defined space.
  - Object must be occupying space for 10 seconds to change occupancy status.
  - Blue rectangle is drawn when vehicles are detected; system & occupancy status updates.

Accomplished Work - Functionality

- Detecting occupancy within a user-defined space.
- Detecting objects within space for validation.
- Movement of space using keystrokes (WASD keys).
- Ability to reset occupancy reference frame (R key).
- Dynamic updates when occupancy status changes in real-time.

Accomplished Work - Frontend

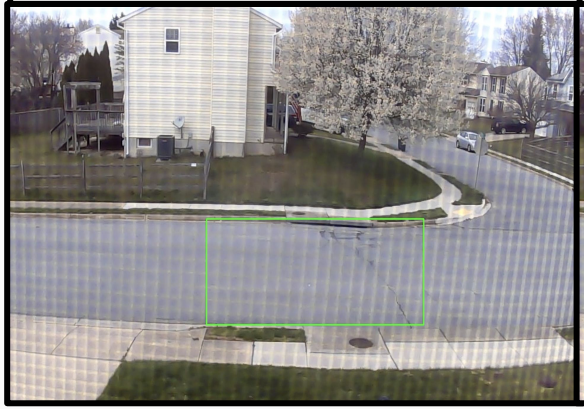
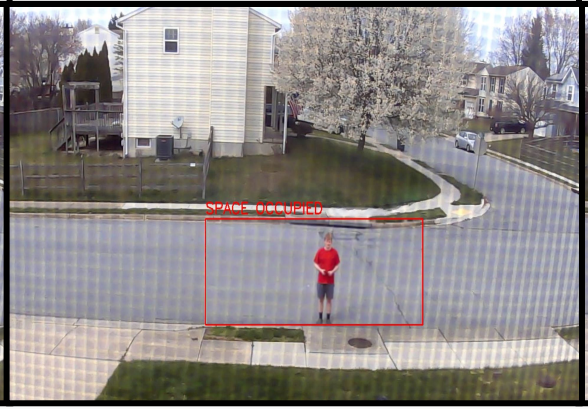
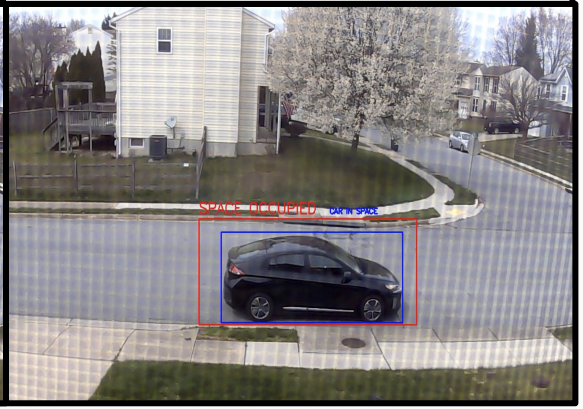
- Pluto connection infrastructure is setup, but not working (yet!).
- Authentication required for connection to Pluto.
- Runs in offline mode if authentication fails.

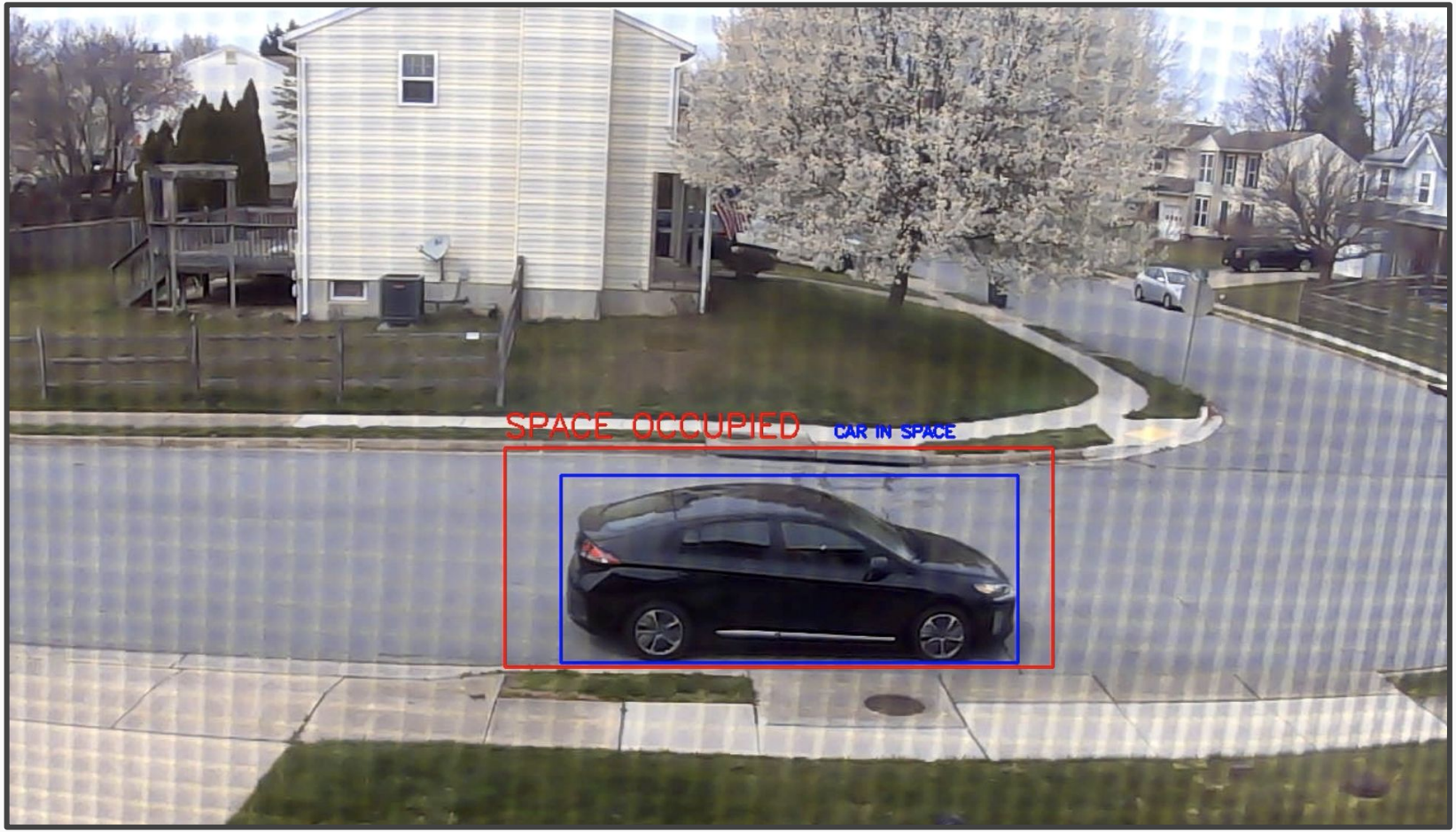
Accomplished Work - Robustness

- Error handling including authentication failures.
- Constant video feed status checks to ensure system accuracy.
- Offline mode is used when having issue connecting to the web server.
- Keystrokes give user control of parking space location on video feed.

Project Status

- Completed all necessary components for back-end video processing

Space Definition & Reset Point	Space Occupancy Detection	Vehicle-in-Space Detection
		


SPACE OCCUPIED
CAR IN SPACE

Technical Takeaways

- Focusing on a single parking space is more feasible for the project scope
- Capturing live video feed performed better than saving images to local FS
- Using pre-trained weights for object detection proved more accurate results
- SFTP can introduce unwanted runtime errors and exceptions

Project Effort

Team Member	Elijah	Subha	Jake	Total
Recorded Time	21:05:00	8:35:00	9:00:00	38:40:00
Efforts	• Backend Application • Started frontend (Pluto connection)	• Web App UI Research and Prototyping	• JavaScript Library and Weather API Research	

Management Tools Used: VS Code, GitHub, Google Sheets, iMessage

Future Project Work

- Generate graphs on system status and time data.
- Collecting time stamps when system status changes.
- Create system error logging system for both frontend and backend.
- Connection of weather API for location weather.