Traffix

D8 - Ankita Chatterjee, Kaitlyn Liu, Zina Zarzycki
18-500 Capstone Design, Spring 2024
Electrical and Computer Engineering Department
Carnegie Mellon University

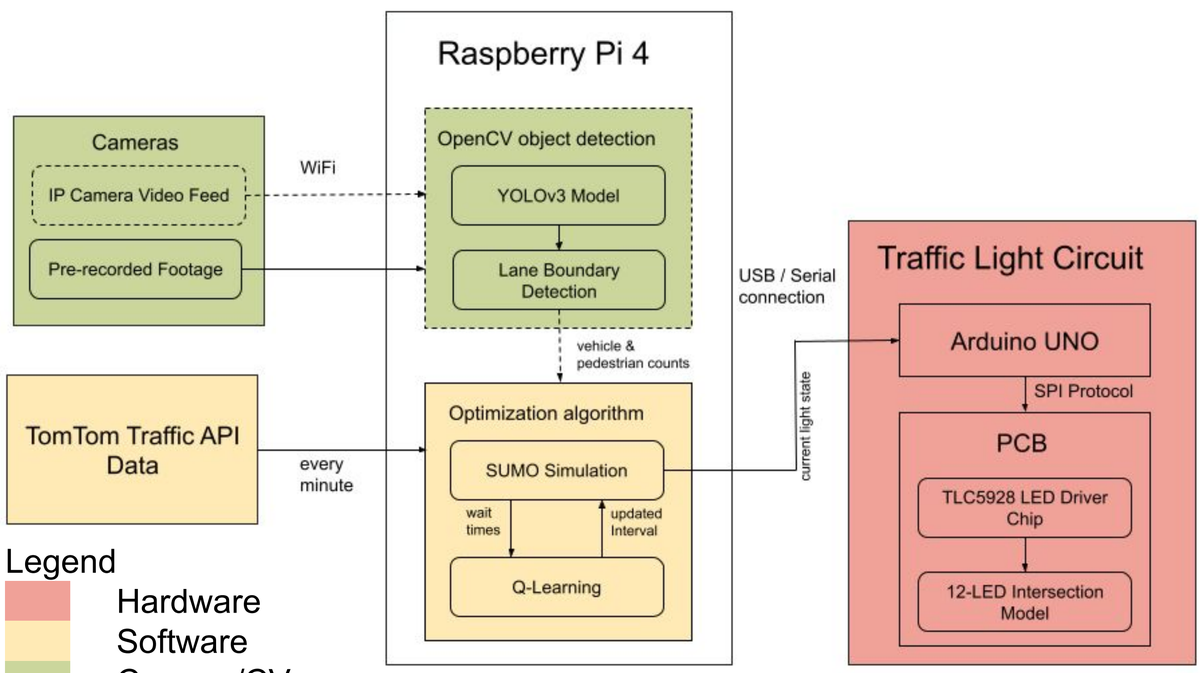
Product Pitch

The average American spends two-and-a-half work weeks in traffic per year. Congestion at intersections fluctuates frequently throughout the course of a day, and most traffic lights cannot anticipate sudden variations in traffic flow. This is especially evident in scenarios where roads are closed and detours are in place or other events that cause drastic changes in traffic density over a short period in time.

Traffix aims to reduce traffic congestion by creating a smart traffic light that leverages machine learning and computer vision to identify cars and pedestrians at intersections and dynamically calculate light interval times using a Q-learning algorithm. This will benefit drivers by reducing wait times in traffic as well as benefit the city by providing a lasting system that increases city efficiency as the online machine learning model adapts to changes in traffic over time.

System Architecture

Our system has 3 main subsystems: the **object detection**, the **optimization algorithm**, and the traffic light **PCB**. The object detection code takes in 4 pre-recorded videos of the intersection and uses the pretrained **YOLOv3** model and additional image processing to determine the number of cars and pedestrians in each lane. The optimization algorithm runs on the Raspberry Pi and uses simulated vehicle and pedestrian counts to determine an optimal light interval, and uses feedback from the **SUMO simulation** to update the **Q-learning** model. The **Arduino** receives the updated light states every second and changes the LEDs on the traffic circuit PCB accordingly. In an ideal implementation, the **Raspberry Pi** would run both the object detection and the optimization algorithm, but because the videos do not reflect the behavior of the simulation (i.e. the light states in real life and our simulation are not the same, so the car behavior will not be the same) we chose not to integrate them and instead demonstrate them separately.



Conclusions & Additional Information

Visit our website!

http://www.ece.cmu.edu/~ece500/projects/S24-teamD8

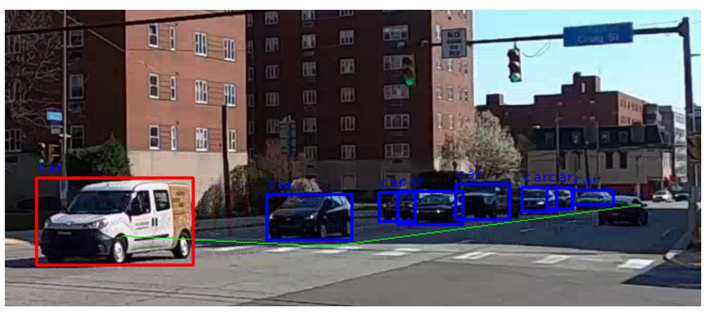
Through our testing, we realized that our optimization algorithm does converge to a fixed light interval, however we saw decreased wait times and more variation in the model output for extreme scenarios where car density was extremely high due to traffic jams. We hope to continue adjusting our product to be more sensitive to changes in the state and support more connected systems across cities rather than at individual lights.

Also, we would have liked to use the live IP camera video feed rather than pre-recorded footage to run the object detection model, and do a better job of integrating the two. As it stands, the object detection model processes frames from all 4 sides of the intersection simultaneously, which ended up being too computationally intensive for the Raspberry Pi 4 to handle (note that we weren't able to meet our latency requirements for that metric.). Other than that, however, our system meets the desired metrics for wait time reduction and complexity handling.

System Description

Object Detection

We used a pretrained YOLOv3 model to detect the total number of vehicles and pedestrians in one frame. The bright green line indicates the determined lane boundary for the frame, and only cars within that boundary are included in the final count.

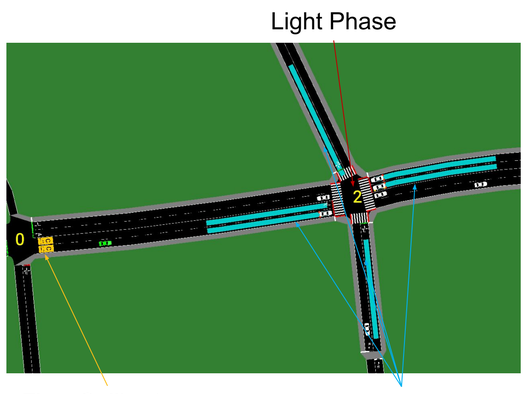


We initially intended on processing IP camera footage instead of pre-recorded videos but had difficulties accessing the IP camera stream from the Raspberry Pi and chose to simplify our demo.

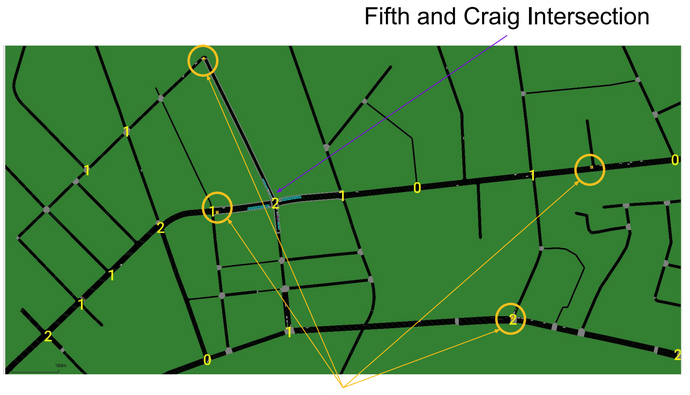
Optimization

We used a deep Q-learning reinforcement learning model to optimize the timing of the traffic light. At a fixed rate, the model outputs how long each side of the intersection should be green.

SUMO Simulation at Fifth and Craig

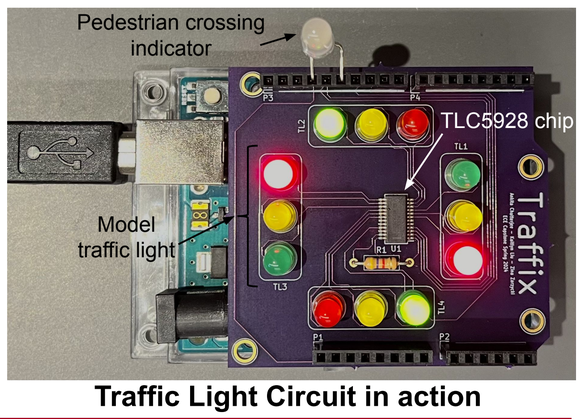


Entire SUMO Simulation



Traffic Light Circuit

To demonstrate our system's ability to react to observed traffic situations, we designed and assembled a custom PCB to model a four-way intersection. In this circuit, the Arduino gets current state information from the RPi, then transmits a corresponding 16-bit data string to the TLC5928 LED driver chip to enable the desired lights.



Traffic Light Circuit in action

System Evaluation

For our optimization testing, we tested with a control using a fixed light timing system at the Fifth and Craig intersection, which we gathered from footage recorded from the actual light, comparing those wait times to wait times while using our Q-learning model.

To test the object detection model, we evaluated the accuracy of our calculated vehicle and pedestrian counts over 100 frames, taken from 4 videos corresponding to each side of the Fifth and Craig intersection. We determined how many cars the model could handle per side by noting the maximum number of vehicles that the model could count accurately.

Metric	Target	Actual
CV Model Accuracy (cars)	90%	80%
CV Model Accuracy (pedestrians)	80%	90%
Optimization Wait Time Reduction (regular conditions)	10%	16.4%
Optimization Wait Time Reduction (stress conditions)	10%	42.2%
CV Model Latency	5s	~12s*
Optimization Model Latency	100ms	~100ms
Complexity Handling	10 cars per side	11 cars per side
Constant LED Forward Current	10 to 15 mA	10.7 mA (using 4.7kΩ resistor)

* Time taken for the Raspberry Pi to process 4 frames at a time. ~4s to process 1 frame alone.