ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

V2V Based Network Cooperative Control System

Team #22

XINWEN ZHU
ZIHAO LI
YUXUAN JIANG
JIAZHEN XU

TA: Xuyang Bai

May 15, 2023

Contents

1 Introduction

1.1 Background

Intersections pose a significant risk to transportation safety, as they account for a vast majority of severe urban traffic accidents. For example, about 43% of all crashes in the United States occur at or near an intersection [1], about 40% of all casualty crashes in Norway occur at junctions, about 33% of crashes in Singapore. Moreover, these numbers kept increasing over the years[2].

This situation primarily arises from the following two reasons: It's a common case the vehicles traveling in orthogonal directions cannot notice each other due to the obstruction of buildings, thus leading to a high risk of collision. Besides, non-motor vehicles and pedestrians are more possible to appear in the blind spot vision of vehicles, since there are usually plenty of them gathering in the intersection. These two facts create difficulties for a single vehicle to observe potential collision and avoid it, no matter what technique it exploits.

Additionally, today, most of the urban intersections are under passive control mechanisms such stop signs, yield signs and traffic lights. Stop signs require vehicles to come to a complete stop, even when there are no other cars at the intersection. This reduces efficiency by causing unnecessary deceleration. According to a very conservative calculation performed by Victor Miller at Stanford University[3], unnecessary traffic stops in the United States can account for 1.2 billion gallon consumption per year, which can satisfy an average American to fill up a 15 gallon tank every other week. Such passive intersection control mechanisms have lead to significant amount of energy waste and call for adaptive control mechanisms.

1.2 System Diagram

To address the **safety** and **efficiency** issues at urban intersections, we propose the V2V-based Network Cooperative Control System (VVCCS). The vehicle-to-vehicle (V2V) communication can help the vehicle get a holistic view of intersection conditions and make intelligent decisions accordingly. Figure 1 showcases an intersection scenario that our VVCCS focuses on.

1.2.1 Initial Design

We present the overview of our system in Figure 2, which consists of four subsystems:

1. **Control Subsystem**. The control subsystem ensures that the vehicle can run at the desired speed, accelerates and decelerates in time. The vehicle will be controlled by commands with the parameter of pulse-width modulation (PWM) and steering angle. Besides, localization is also implemented via the control subsystem by estimating the motor state to infer the position of the vehicle.

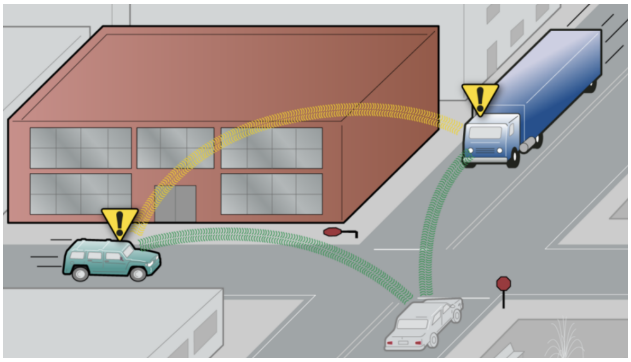

Figure 1: intersection scenario

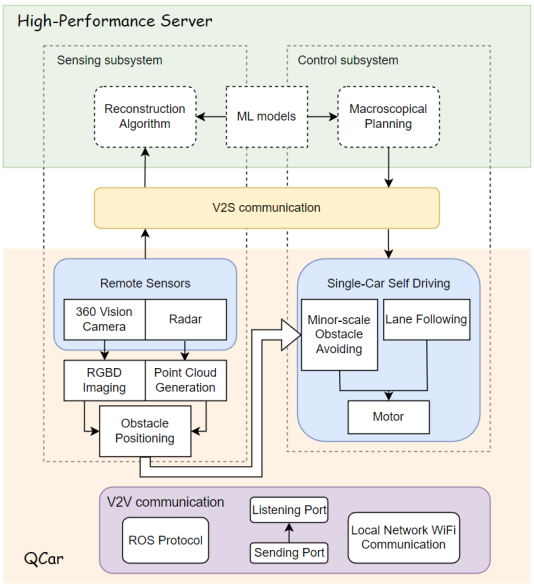

Figure 2: system overview

2. **Information Sensing and Fusion Subsystem**. We use cameras and the Lidar to sense the environment. Approaching vehicles and pedestrians at the intersection will be recognized and tracked. To realize a better precision, information from cameras and Lidar will be merged before being sent to other vehicles through V2V network.

3. **Communication Subsystem**. All vehicles equipped with V2V communication technology will be able to share their current state, including location, velocity, acceleration, and heading, with each other in real-time. This will enable each vehicle to have a holistic view of the intersection's traffic condition and make informed decisions to avoid potential accidents.

4. **Collision Avoidance**. The avoidance algorithm running on both the server and vehicles will analyze the data received from the V2V communication and intersection approach recognition subsystems and make decisions on the best course of action to avoid potential collisions. Information like vehicle velocity, distance, and direction will be considered to ensure both safety and efficiency.

1.2.2 Final Design

During this semester, we made several block-level changes based on our study and progress. The final version overview of our design is shwon in Figure 3. We respectively ellaborate the changes made to each subsystem.

1. **Control Subsystem**. We added a feedback term of detected motor speed to construct a closed loop for controlling more accurately (the arrow from "Motor" to "Vehicle Positioning").

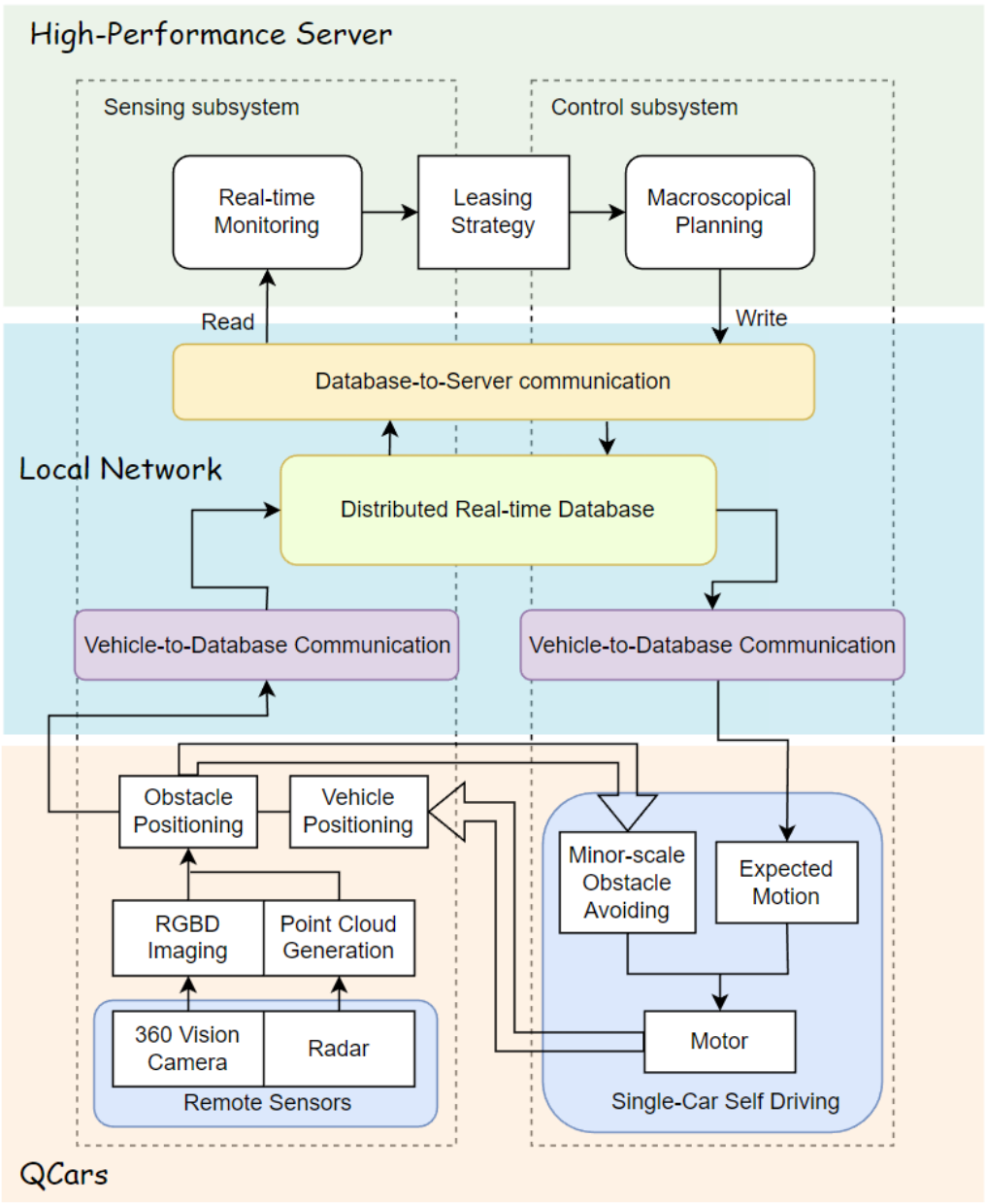

Figure 3: Final Design

2. **Information Sensing and Fusion Subsystem**. We combined detected obstacle positioning and calculated vehicle positioning together.
3. **Communication Subsystem**. Instead of direct communication, we set up a distributed database and conducted read-write style communication.
4. **Collision Avoidance**. We changed the planned ML models into leasing strategy because of the real-time response requirement and the time limit.

1.3 Performance Requirement

To ensure that our proposed solution is effective and efficient, we have established the following high-level performance requirements:

1. **Control Accuracy**: The control subsystem must performs immediate response ($< 0.1s$) to the signal generated from top-level algorithm. The system also requires the vehicle to go straight (deviation angle < 0.005rad), meaning that the lateral deviation should be less than 1 centimeter after running for 2 meters.
2. **Object detection**: The speed and success rate are major consideration of the object detection. Both slow and unsuccessful detection may cause failure of collision avoidance, hence harming the effectiveness of our control system. The vision-based object detection system must achieve a minimum of 90% success rate for each sample and the algorithm should finish processing 10 frames per second in detecting other vehicles and pedestrians at intersections.
3. **Collision avoidance** : The Collision avoidance algorithm must possess 100% accuracy. The system must achieve a minimum of 99% success rate in simulating collision avoidance at intersections, including some wilful emergencies.
5. **Communication Performance**: The bandwidth and latency are two major factors that influence the efficiency of our system. The bandwidth of communication technology will restrict the package size and information amount we want to transfer, while the latency should be considered when designing the collision avoidance algorithm. We quantitatively assess the condition of laboratory network and make decisions accordingly, which will be discussed in the communication section.
5. **Energy efficiency**: The overall energy consumption of the system must be lower than the energy consumption required by traditional traffic control mechanisms, such as traffic lights and stop signs.

By meeting these high-level requirements, we can ensure that our proposed solution addresses the safety and efficiency challenges at urban intersections effectively and sustainably.

2 Design

Our team use the Quanser Car[4] (Qcar) as the experimental car to finish the design. The Qcar (Figure 4) is equipped with many sensors, such as a LIDAR, a RGBD camera, and two CSI cameras on the left and right side. Those sensors can capture detailed information about the environment. Qcar also has a on-board GPU to support necessary computation for information processing. Besides, we make a detailed investigation on the physical characteristic of Qcar and present it in Appendix A.

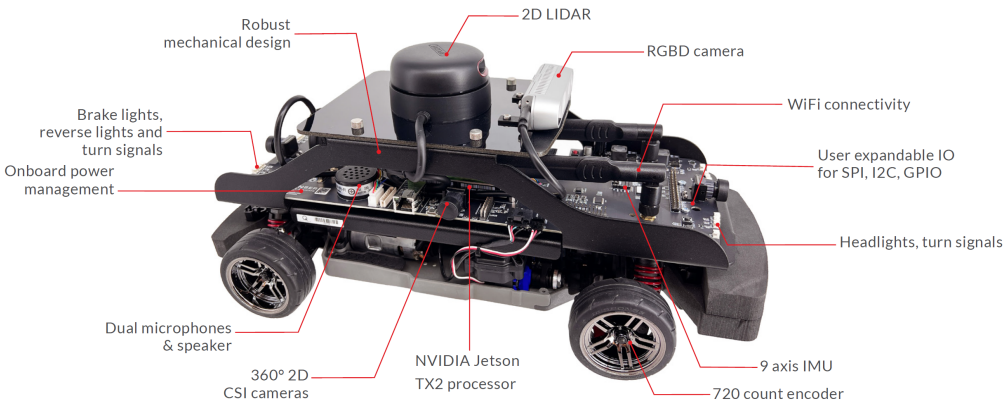

Figure 4: Qcar Diagram

To your best understanding, we will describe our design procedure, design details and corresponding verification for in each subsection, respectively.

2.1 Software Stack

In this section, We will illustrate the software stack of VVCCS on the vehicle, including the top-level design considerations, workflow of the program, and specific functionality of each block. Figure 5 shows the overview of software components.

In the diagram, there are three main blocks (orange, blue, and green) and red arrows between them. The red arrows indicate inter-block data transfers, as main blocks continuously share data via corresponding function calls. The orange block refers to Python files that read and process the raw data from the hardware (like Lidar, cameras, and motors). We use Python here because the Qcar itself provides some basic hardware drivers in Python. The green block refers an environment pool implemented in *etcd*, a distributed and reliable database, where we store all observed objects. The blue block refers to the main function of VVCCS. We realize it in Go, since it can both call the Python API and easily interact with etcd. After launching, all three main blocks will be initialized and will run concurrently.

The main function in Go principally contains a loop with three code blocks: Perception, Decision-Making, and Control. The perception block polls the Python interface to get

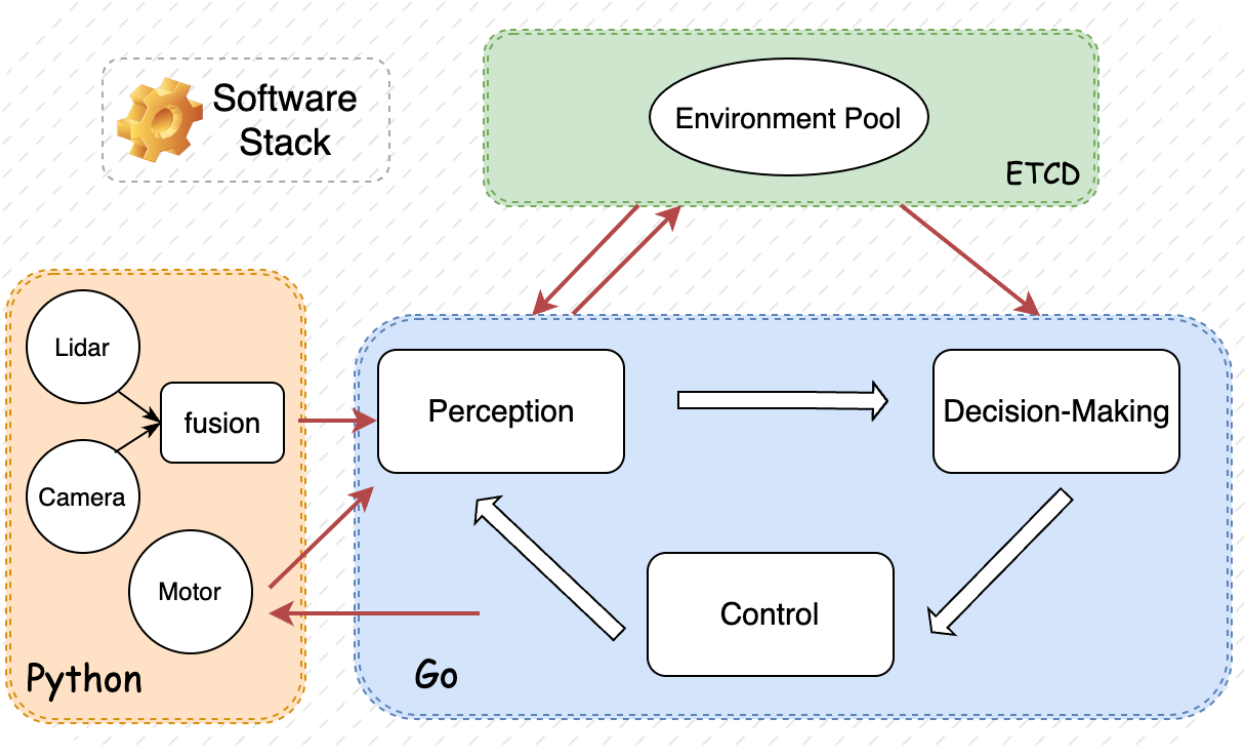

Figure 5: Software Stack

data. The system will then obtain the state of observed surrounding objects from the Lidar and Camera after a information fusion module, and the state of itself from the motor. Thereafter, the perception block will create new objects, delete outdated objects, and update the state of objects that are under tracking in the etcd. The decision-making block runs the whole collision avoidance algorithm with stored data in etcd, and output desired speed to the control block to enforce it. After finishing all three blocks, our system will wait for a certain time before entering the next loop so that it can stay synchronized with the sampling and information fusion module in Python.

The design of software stack is the last step to finish VVCCS, so the verification is relatively straightforward. As long as the system works, it will prove that our design satisfy all requirements.

2.2 Control subsystem

This section shows the design and implementation details of the control subsystem of VVCCS. In order to success, the control subsystem must fulfill the following three requirements:

1. The vehicle goes straight.
2. The vehicle can run at given speed.
3. The vehicle has the ability to measure the location of itself, so that the relative po-

sition of different vehicles can be calculated after they receive messages from each other.

At first, we intuitively think that each Qcar will definitely go straight. But actually the deviation problem is severe because the diameter of the wheels on the two sides are not perfectly equal, and the suspension cannot maintain totally horizontal. Thus, we measured the steering parameter for two Qcars, which are -0.066 and -0.041, respectively.

As the motor of Qcar is controlled by PWM instead of voltage change, we need to build a map from the duty cycle to the actual speed. After the sampling and analysis, we find that their relation can be expressed as a linear function shown in Equation 1

$$dutyCycle = k * v_{target},\ k = 0.1 \tag{1}$$

However, the Qcar takes too much time to achieve the desired speed when we want to enforce the collision avoidance algorithm in such control system. Hence we add feedback terms to realize faster convergence . In Equation 3, we introduce an proportional error and an integral error so that the system can provide additional power to approach the desired speed. The intuition behind it is that the proportional error can provide rapidly respond to a change of desired speed and the integral error will augment the correction as long as the difference exists. It successfully fulfill the requirement 2) for the control subsystem.

$$e(k) = v_{target} - v_{current} \tag{2}$$

$$dutyCycle = k_{ff} * v_{target} + (k_p * e(k) + k_i \sum_{i=0}^{k} e(i))) \tag{3}$$

Lastly, we need to handle the localization issue since Qcar is not equipped with a GPS module. We proposed two alternative plans to tackle this problem. The first one is to set up several signs and make each vehicle to measure its relative position against the sign, as we have a sensing subsystem can recognize those objects and report the location. Nonetheless, our test shows that the sensing subsystem fails to recognize the sign as the vehicle goes away. The sign will become overly small in the camera, to the extent that the sensing algorithm will ignore it. This problem is irreparable due to the restriction of the processor capability of Qcar.

Therefore, we turn to another plan, which requires the Qcar to measure the distance it travelled by monitoring the state of motor. Once the Qcar can estimate its real-time speed with the state of motor, we can integrate the speed to calculate the distance. Additionally, we hardcode the initial position of Qcar into the system and always run it from there, so that the Qcar can calculated its location accordingly. As we can only monitor the count of motor's spin, we deduce following formula (Equation 6) to further compute the speed of vehicle:

$$k_{ps} = ((13 * 19)/(70 * 37)) \tag{4}$$

$$k_{cr} = (1/720/4) * 2\pi \tag{5}$$

$$V_{car} = k_{ps} * k_{cr} * r_{wheel} * V_{motor} \tag{6}$$

k_{ps} refers to the ratio of pinion to spur. k_{cr} refers to the ratios of counts to rad. V_{motor} refers to the encoder speed in counts/s. r_{wheel} refers to the wheel radius. All parameters are listed in Qcar's product manual.

We verify the effectiveness of control subsystem by conducting two experiments:

1. We make the Qcar run for a preset distance (D), and measure its lateral deviation (d_l)as well as the actual distance (d_a).
2. We make the Qcar run in the speed v_1 and output its real-time speed to the console. We then change the speed to v_2 by commands at t_1 and record the first time when the output speed goes below v_2 as t_2. The time different $\delta_t = t_2 - t_1$ will be used to judge the performance of our control algorithm.

D (m)	d_l (cm)	d_a (m)
1	0.41	1.004
1.5	0.72	1.508
2	0.93	1.993
2.5	1.30	2.515
3	1.68	3.019

v_1 (m/2)	v_2 (m/s)	δ_t (s)
0.3	0.1	0.007
0.3	0.5	0.08
0.3	1	0.23
1	0.3	0.012
1	2	0.39

Figure 6: Experiment Results

Figure 6 illustrates that the Qcar's response time to speed change is negligible while both lateral deviation and distance measurement satisfy the requirement discussed in section 1.3.

2.3 Sensing subsystem

In this section, we will present a detailed description on the sensing subsystem, including its primary components, information processing procedure, and the refinement for better performance.

2.3.1 Primary Components

Camera
Qcar contains a Intel D435 RGBD Camera and a 360°2D CSI (Camera Serial Interface) Camera system, as shown in Figure 4. There are four 8MP 2D CSI cameras (Figure 7) at the front, left, rear and right side of the vehicle. Each camera has a wide-angle lens providing up to 160° Horizontal-FOV (field of view) and 120° Vertical-FOV. The corresponding blind-spots have been shown below in Figure 8

At first, we plan to use the D435 RGBD Camera because it can measure the depth of objects as well as shooting the photos. However, it's hard for us to run an real-time object

Figure 7: CSI Camera

Figure 8: Blind spots of camera

detection algorithm on the output, and the only one D435 RGBD Camera cannot generate a holistic view of surroundings of the vehicle. So we abandon this plan and turn to adopt the CSI cameras for sampling.

LIDAR

The Qcar platform provides RPLIDAR A2M8, an enhanced version of 2D laser range scan- ner(LIDAR). It can perform 2D 360 degree scan within a 12-meter range(8-meter range of A2M8-R3 and the belowing models). It takes up to 8000 samples of laser ranging per second with high rotation speed. The typical scanning frequency of the RPLIDAR A2 is 10hz (600rpm). Under this condition, the resolution will be 0.45°. During every ranging process, the RPLIDAR emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected. Figure 9 illustrates the working principle and provides an example output of RPLIDAR.

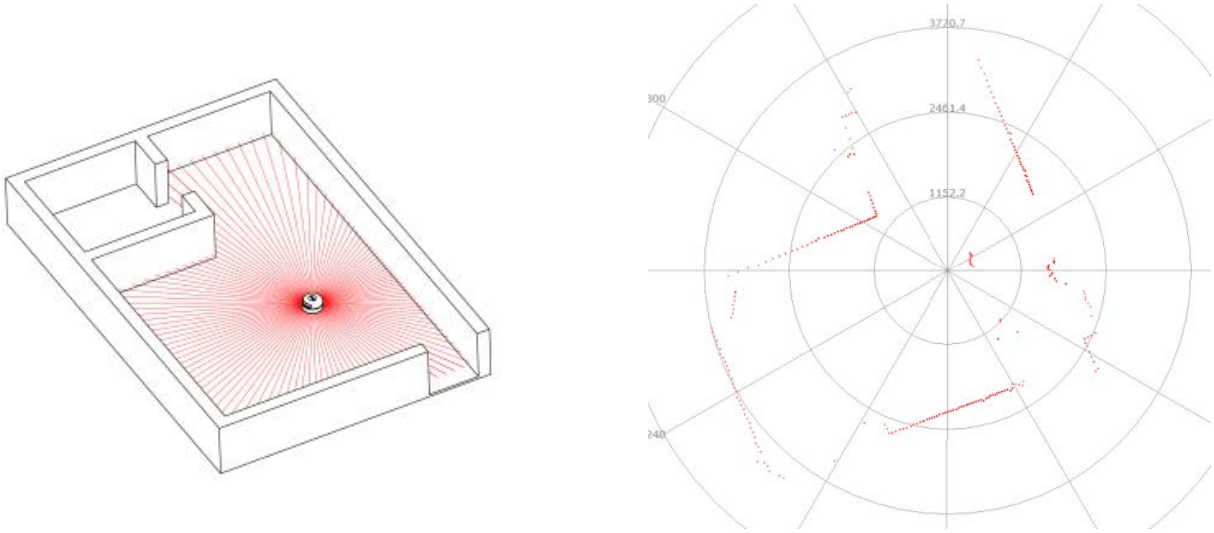

Figure 9: RPLIDAR mechanism

2.3.2 Information Processing

A basic requirement of our sensing system is to detect what are the surrounding objects and their positions. Using cameras and RPLIDAR, we get these two kinds of information respectively. We then fuse such information together (1-1 correspondance). In this part we introduce our work on getting data.

Object Detection Model

For object detection, the Qcar hardware supports 256 CUDA Core NVIDIA Pascal™ GPU architecture,1.3 TFLOPS (FP16) NVIDIA® Jetson™ TX2. The AI performance of Jetson TX2 series is 1.33 TFLOPS. The four cameras support four initial image data, from which we construct one image containing these four initial images as sub-images. We then input the combined image into the Yolo5 object detection model in real-time. An example figure is shown in Figure 10.

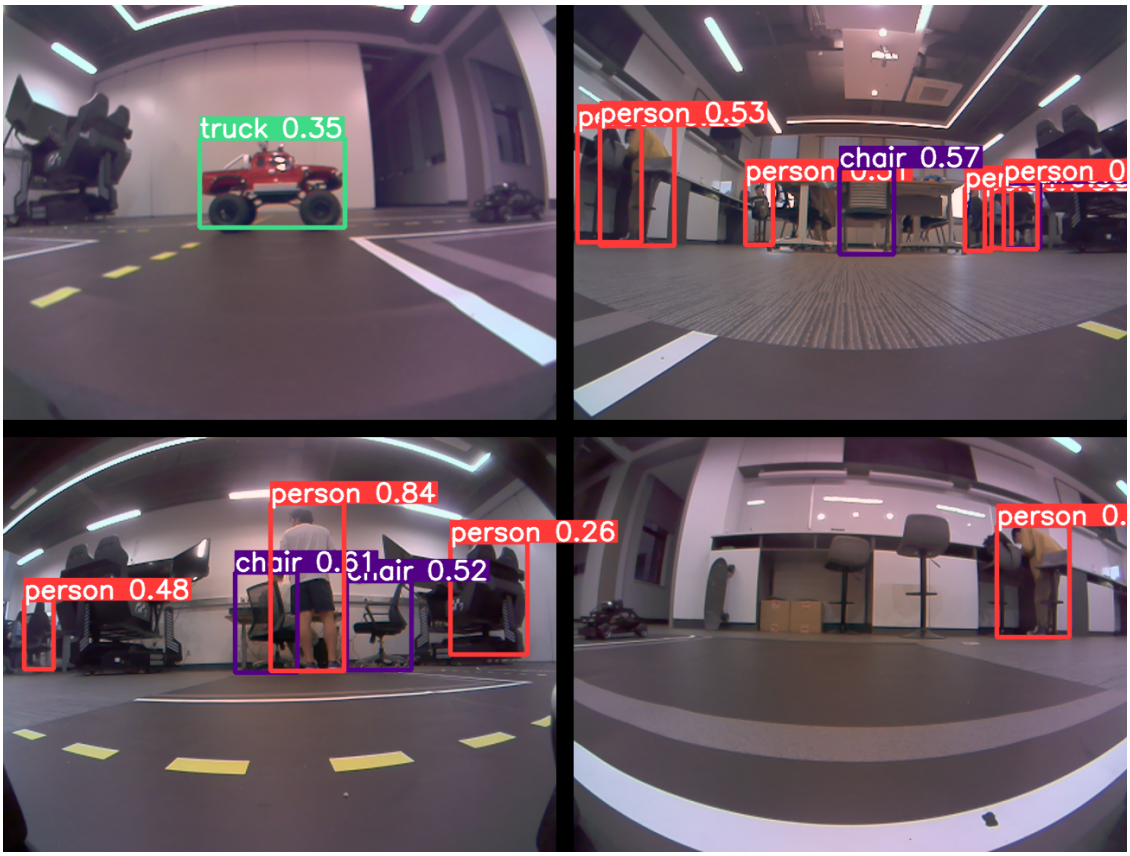

Figure 10: Object Tracking

We applied multiple models and chose the one that has the best performance in our domain. We also tuned the hyper-parameters as 10 frame per second to achieve our real-time requirement.

Image Coordinates to Angle Conversion
To get the accurate position of surrounding cars, we need to first get the relationship between image coordinates of objects and their angle to the Qcar frame. After experiment, the raw data is like (Figure 11):

After curve fitting the data in cubic equation (Figure 12):

$$\textbf{angle} = a * \textbf{pixel}^3 + b * \textbf{pixel}^2 + c * \textbf{pixel} + \textbf{bias} \tag{7}$$

where $a = 1.08838458e-07$, $b = -5.36308439e-06$, $c = 1.83988097e-01$, $bias = 4.00103535e-01$.

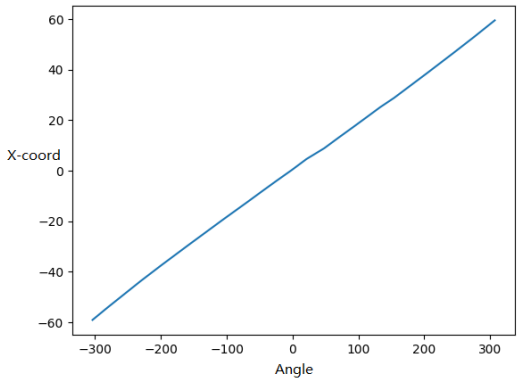

Figure 11: Raw data

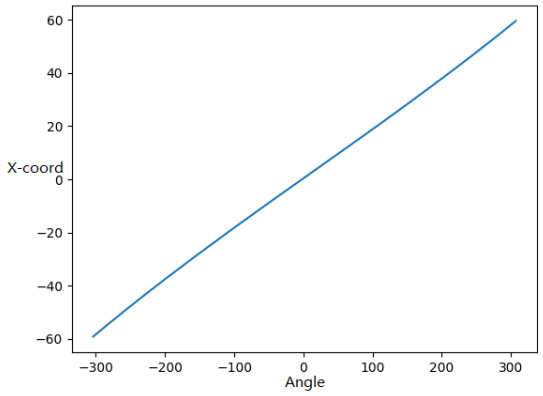

Figure 12: Curve fit

Since a and b are very small and the bias is mostly resulted from the inaccuracy of measuring the offset angle, so we can simply assume there is a linear relationship between image coordinate of object and angle to Qcar in CSI camera with $\textbf{angle} = 0.1840 * \textbf{pixel} + \textbf{bias}$. We measured and fitted the pixel-to-angle correspondence and angle-to-pixel correspondence. The fitted curve can be approximated linearly in an accurate scale.

Data Fusion
The angle of surrounding cars to the Qcar can be obtained by processing images from four cameras. At meantime, LIDAR will provide a point cloud map containing angle and distance of each point. By extracting points in the range of angles obtained before, we can get the distance to the surrounding car and finally get the accurate relative position by calculating through angle and distance.

2.3.3 Performance Optimization

Multi-thread Optimization
Each loop of our perception system includes four steps: fetching data from each sensor (camera, LIDAR, motor encoder), running object detection model, fusing prediction from model with data from sensor, sending command to control system. However, running such a loop serially will spend about 130ms, and it will be impossible for us to run the system above 10Hz.
When fetching data from sensor, most time is wasted to wait for system API and we mostly use CPU resources. When running object detection model, GPU resources is mostly used. Before fusing data, prediction from object detection model must be prepared. So that we can run object detection model and fusing algorithm in the main loop with some threads for each sensor to fetch their data independently.

Elimination for Duplicates
In some cases, surrounding objects will be detected by two adjacent cameras at the same time. If they are simply treat as two objects and calculate their relative position separately, the result will be inaccurate. So that we can extract their label box from the prediction list

of two adjacent cameras and merge them into one large box.

On the other hand, surrounding objects may be detected as two due to non-max-suppression algorithm of object detection model. In our code, if two detected objects' relative position are too close, they will be merged and there will be only one object finally.

Absolute Position Calculation
The position of surrounding objects in absolute coordinate will be calculated through Qcar's own absolute position, Qcar's heading and relative position of surrounding objects.

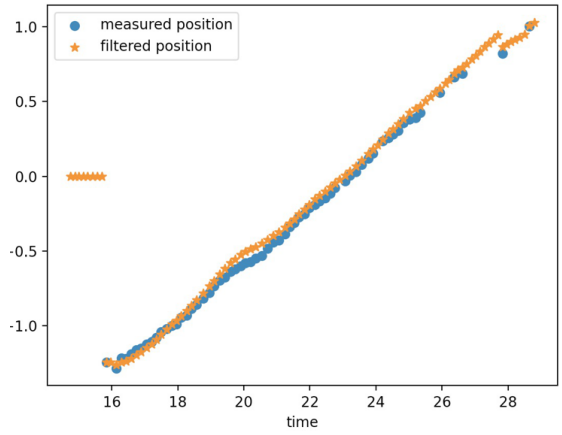

Figure 13: Kalman Filter Result

```python
# merge cars' image on the edge of each camera
new_cars_angle = []
for car_1 in cars_angle:
    if car_1[3] is not None:
        for car_2 in cars_angle:
            car_matched = False
            if car_2[3] is not None and car_2[3] == car_1[2] and car_2[2] == car_1[3]:
                car_matched = True
                if car_1[2] < car_2[2]:
                    if car_1[2] != 0 or car_2[2] != 3:
                        new_cars_angle.append((car_1[0], car_2[1], car_1[2], car_2[2]))
                        break
                    else:
                        new_cars_angle.append((car_2[0]+2*np.pi, car_1[1], car_2[2], car_1[2]))
                        break
                elif car_1[2] > car_2[2]:
                    if car_1[2] != 3 or car_2[2] != 0:
                        new_cars_angle.append((car_2[0], car_1[1], car_2[2], car_1[2]))
                        break
                    else:
                        new_cars_angle.append((car_1[0]+2*np.pi, car_2[1], car_1[2], car_2[2]))
                        break
        if not car_matched:
            new_cars_angle.append((car_1[0], car_1[1], car_1[2], None))
    else:
        new_cars_angle.append(car_1)
```

Figure 14: Inter-camera Data Processing

2.3.4 Data Smoothing

Kalman Filter
Kalman Filter is an algorithm to compute a smoothed time-evolving data given the measured non-smooth data. An example of KF output time-varying position (y-axis) is shown in Figure 13, given the input as the measured time-varying noisy position.

Inter-camera Data Processing
The surrounding object that is detected by two adjacent cameras at the same time will be merged to improve the accuracy. A code snippet handling this part is shown in Figure 14. For the full code, please refer to the appendix.

2.3.5 Verification

To meet the demanding requirements of our perception subsystem, accuracy and real-time capabilities take center stage, as highlighted in section 1.3. The object detection model exhibits an effective working range of approximately 2 meters, boasting an impressive accuracy rate exceeding 90%. Enhancing the computational power of Qcar and employing cameras with higher resolution can significantly enhance the performance of the object detection model. Considering the real-world scale, our system must efficiently detect cars within a range of 10 meters, ensuring comprehensive coverage. However, in

the more localized Qcar scale, the range narrows down to approximately 1 meter, necessitating the assistance of LIDAR, which excels in detecting objects within a 4-meter range. As for the real-time requirement, our perception subsystem surpasses expectations, as it boasts a sample rate exceeding 10 times per second, comfortably meeting our requirements for real-time processing and analysis.

2.4 Communication subsystem

In this section, we present the communication subsystem of VVCCS. To implement a real-time communication while supporting multi-object tracking, we have purposed two communication methods. We will illustrate them separately and explain the reason of our final choice.

2.4.1 Method 1 - ROS communication protocol

The Robot Operating System (ROS) [5]is a distributed communication mechanism built on TCP/UDP, and it processes communication at the granularity of process. In the ROS communication framework, every process is regarded as a *node* (r_node), and messages are passed via logic channels called *topic*s. When a node release a topic, all other nodes subscribe to him will receive that topic. Therefore, ROS satisfies our requirement for communication.

In the original plan, ROS is used to build both low-level communication (between cameras, Lidar, or other devices) and and the high-level communication (V2V). We first need to launch a master r_node in the Ubuntu PC, and register the central r_node of the Qcars within local WIFI network to achieve V2V communication. For the low-level part, each device will have its own node as the port to send data to the central r_node of Qcar. However, when approaching the high-level communication, we discover that it is infeasible to make the Qcar central node directly communicate in the ROS network because ROS provides weak functionality to support communication between two embedded system equipped with it. Consequently, we modify the design and construct a stream between Qcars and the Ubuntu PC. More specifically, we implement a server and a client program using provided python library "Quanser" and run them at the same time to achieve information communication. Figure 15 is the architecture for the ROS communication method.

2.4.2 Method 2 - etcd Database

etcd [6] is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines. etcd exploits the Raft consistency algorithm to coordinates nodes in the cluster. The algorithm elects a master node as leader, who is responsible for synchronization and distribution. When the leader fails, the cluster automatically selects another node as the lead to synchronize the data. Hence, etcd is highly resistant to potential client or communication failure and guarantees the robustness of VVCCS.

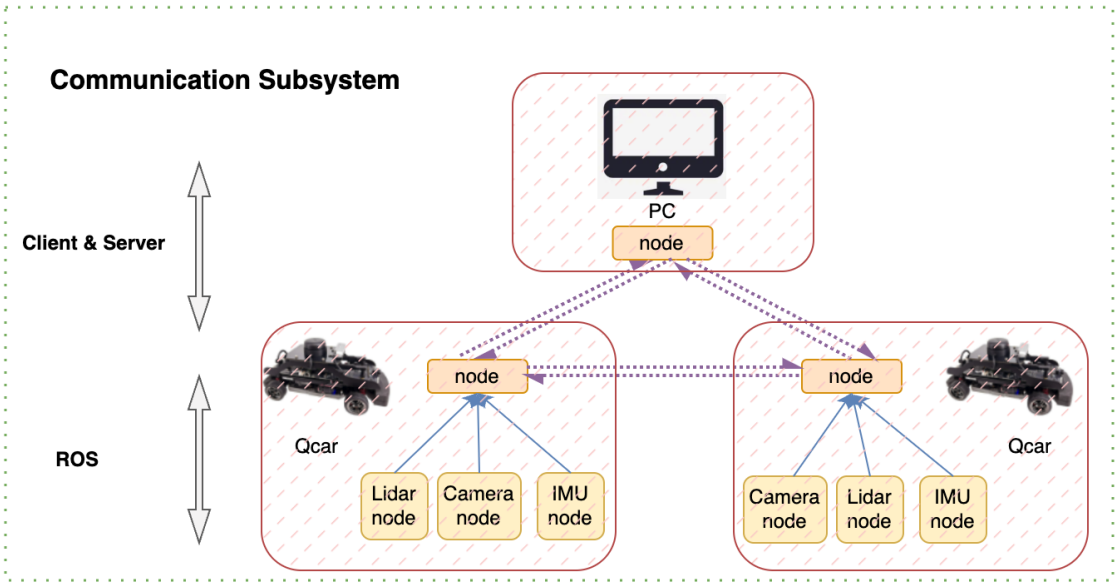

Figure 15: ROS Communication

To implement the communication between vehicles while isolate private data, we construct the following data structure in etcd, as shown in Figure 16. For each vehicle, there are two fields, *state* and *surrounding*, recording the state of vehicle itself and detected objects, respectively. As etcd provides the functionality of synchronization, the vehicle only need to update relevant information to its fields. When the vehicle need to make decision, it will fetch data from all *surrounding* fields in the etcd, thereby realizing the interaction with other vehicles.

Our team finally choose the method 2 as the final design, because it has better scalability and robustness. In the real-world case, many vehicles may come and leave the intersection dynamically, so the system need to link newly joined nodes. The stream used in method 1 requires manual configuration and operation on the console, thus fail to support more vehicles beyond the first setup. Moreover, accidental failure of vehicle clients may directly harm other processes, like making them wait for more time, in the design of method 1. We intend to decouple the communication and planning module for the vehicle, which is the advantage of method 2.

2.4.3 Verification

As mentioned in section 1.3, *bandwidth* and *latency* are critical criteria for designing the communication module. We make reasonable assumption that there are at most 50 cars at the intersection, and synchronization latency less than 100ms is regarded acceptable.

In our experiment case, upload and download speed can maintain stable at 30MB/s in the WIFI6 local area network. For the communication protocol using method 2, each vehicle updates to etcd by sending a 4KB package every 100ms. A full synchronization among all vehicles takes 2000KB/s, while occasional location updates should take 1/10 of the

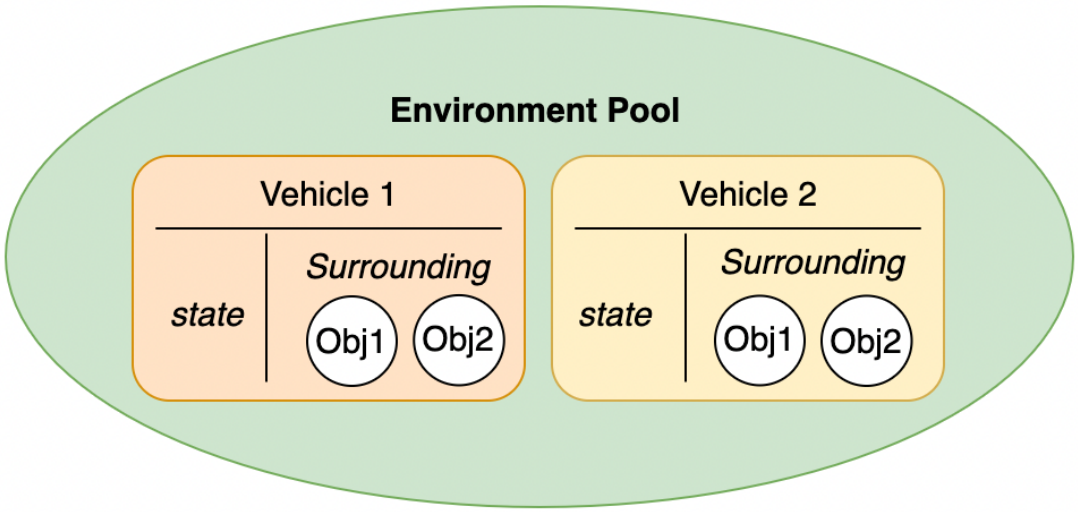

Figure 16: data base structure

maximum bandwidth, 200KB/s. Therefore, the estimated bandwidth of our method in the real intersection scenario is approximately 2MB/s, which is much less than 30MB/s bandwidth provided by the laboratory network. LTE or 5G technology can also serve our use case. On the other hand, the etcd official benchmark presents that reading one single key after putting has the 90th Percentile Latency of 8.6 ms on 64 clients, which suffices to support VVCCS.

2.5 Collision Avoidance algorithm

This section presents our innovative collision avoidance algorithm. We will delineate the input and output of the algorithm, and expound on how it ensures efficiency and safety in scenarios involving both V2V and non-V2V vehicles.

2.5.1 Algorithm Specification

Our algorithm incorporates two subsystems: the lease-based scheduling subsystem and the control subsystem. The former facilitates efficient and timely scheduling for each vehicle aiming to traverse the intersection, while the latter governs each vehicle's motor to adjust speed and direction in alignment with the schedule.

The scheduling algorithm employs a series of discrete snapshots capturing the intersection state at given timestamps. Each snapshot records the speed, location, type, and a unique identifier for every traffic participant. From this data, a lease, delineating the temporal duration of a vehicle's intersection occupancy, is created for each vehicle. A lease can be extended, cancelled, or reapplied in response to unpredictable circumstances.

The control subsystem evaluates the lease assigned to each vehicle, modulating the vehicle's speed to adhere to the lease terms. In essence, the algorithm processes intersection

snapshots as input to produce corresponding vehicle movements as output.

2.5.2 Lease-based Scheduling Subsystem

This subsection provides a comprehensive understanding of the elegance, simplicity, and adaptability of our solution. The ingenuity of our design stems from its ability to strike a balance between efficiency, safety, and timeliness depending on specific application scenarios and computational resources.

Our design hinges on the proven and effective FIFO (First In First Out) algorithm [7] used for obstacle avoidance.

2.5.3 The Lock-based Algorithm

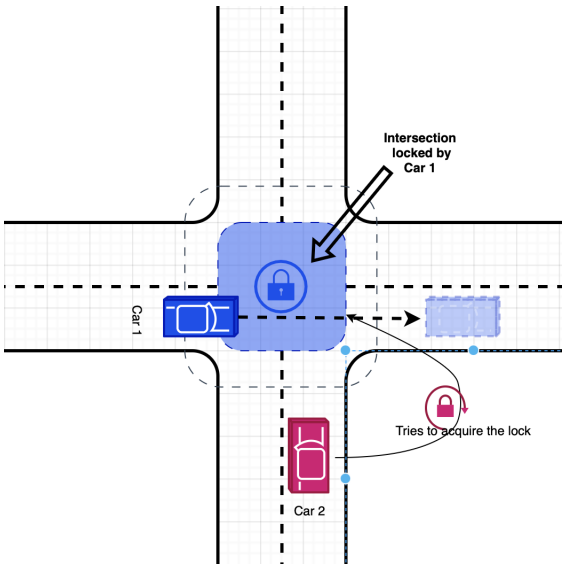

Figure 17: FIFO Algorithm

Our design exploits the highly effective FIFO (First In First Out) algorithm [8] for collision avoidance. This algorithm metaphorically treats the intersection as a computer science lock. Vehicles attempt to acquire the lock before entering the intersection, delaying their entry if the lock is occupied (Figure 18). This one-at-a-time entry policy assures absolute safety.

However, this straightforward approach presents a major drawback: the absence of scheduling capabilities, resulting in safety and efficiency issues. For instance:

- **Abrupt halts:** Vehicles cannot anticipate when a lock might be obtained by others, leading to sudden stops or reduced efficiency.
- **Efficiency conundrums:** Without advance knowledge of lock release, vehicles can't adjust their speed to seamlessly traverse the intersection immediately upon lock availability. This issue exacerbates traffic congestion under heavy traffic conditions.

The root of these issues lies in the algorithm's over-cautious modeling of the intersection.

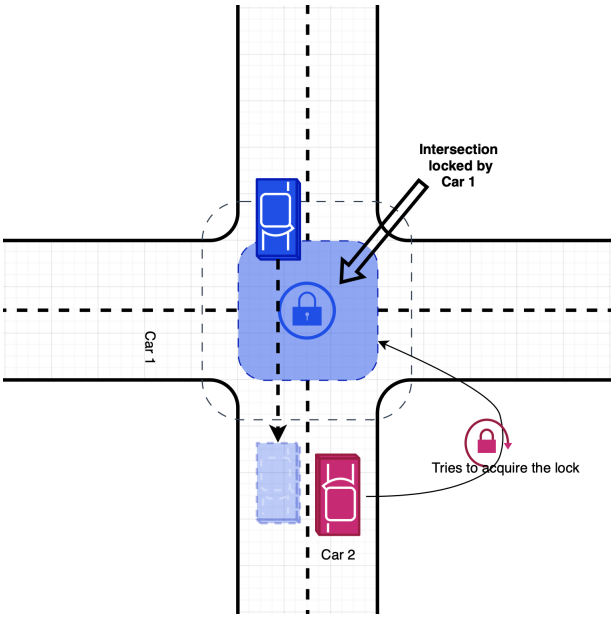

Figure 18: FIFO Algorithm, low efficiency

2.5.4 Lease-based Algorithm

The drawbacks of the FIFO algorithm stem from its insufficiency of information in the lock. We suggest integrating a concept from distributed systems - the lease - to augment the lock's information, thus enabling proactive, intelligent decision-making by traffic participants.

A lease is akin to a lock, supplemented with a conservative estimation of the duration when a traffic participant is expected to occupy a block. Traffic participants holding currently active leases are permitted to enter the intersections. Safety is guaranteed by ensuring leases do not overlap in time for conflicting paths in the intersection.

This lease-based approach grants a temporal window of safe intersection traversal to each vehicle, allowing them to adjust their speed in anticipation of their assigned lease, thereby eliminating sudden halts. Simultaneously, by reserving a time window rather than the entire intersection, it enhances intersection usage, improving efficiency.

In a nutshell, every traffic participants action will be divided into three phases depending on their location.

- Planning (before crossing the intersection)
- Crossing (inside the intersection)
- Post-crossing (after crossing the intersection)

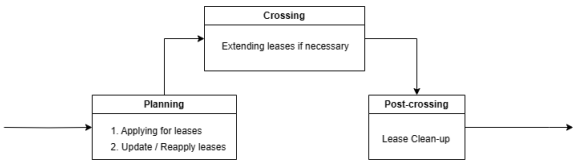

Figure 19: Lease Management, the State Machine

The Planning Phase In this phase, traffic participants will have two kinds of actions, depending on whether it has made an lease or not. Every traffic participant starts with no lease. To apply for a lease, they must follow these steps:

- Estimate the expected time of entering and leaving the intersection area.
- Check if there are any conflicting leases.
- If there are no conflicting leases, register its lease into etcd, using the expected time.
- Else, postpone its lease to the next available slot and register the lease into etcd.

These steps guarantee no two leases can overlap at the applying phase. After a lease has been acquired, the traffic participant should constantly check the following:

- If the current lease can be bring forward? If yes, bring the lease forward to the first available slot. This step is necessary as sometimes, a previous lease can get cancelled. In this case, we want to actively check if a lease can be put forward for efficiency concerns.
- Check if the current lease is still possible to satisfy within the car's mechanical capabilities. If it is impossible to catch up with a lease anymore or the lease has expired, we want to cancel the lease and reapply the lease.

The Crossing Phase In this phase, traffic participants mainly do the following for lease management:

- Check if its lease is about to expire. If yes, extend the lease and postponing other participants' leases if necessary, to avoid other participants from entering the intersection before you leaves.

The Post-crossing Phase In this phase, traffic participants mainly do the following for lease management:

- Cancel the lease if it is still active. A lease might still be active after the participant has left the intersections because of many factors such as conservative time prediction. We want to early-cancel the lease so that other traffic participants can bring their lease forward.

Managing Non-V2V Traffic Participants Not all traffic participants are equipped with V2V communication capabilities and as such, they may be unable to apply for leases

autonomously. To navigate this issue, we assign the responsibility of creating leases for these participants to the V2V participants that detect them. However, the intentions of non-V2V participants are opaque to V2V participants, thus their entry and exit times are predicted conservatively. In the event of a conflict, we prioritize non-V2V leases by postponing V2V leases instead. This strategy is implemented to minimize the impact of unpredictability from non-V2V participants on the safety of the overall system.

2.5.5 Enforcement

Once the lease for each vehicle is assigned, the control subsystem is responsible for ensuring each vehicle adheres to its lease.

The vehicle control subsystem acts as the intermediary between the lease-based scheduling subsystem and the physical layer of vehicle motors, regulating speed and trajectory to meet the scheduling requirements. It receives the assigned lease and the vehicle's current state (speed, location, etc.) as inputs, and then outputs the required speed adjustments and trajectory plans.

- **Planning:** If there is not a lease, the traffic participants should keep going at its current speed. If there is a lease, the participant change its speed according the requirement of the lease. If the participant is about to arrive at the intersection but still does not have a lease available, it stops until the leasing system makes a lease.
- **Crossing:** Keep its speed at the advised speed (often set by the government), stop if the current lease is preempted by a non-V2V traffic participant.
- **Post-crossing:** Keep its speed at the advised speed (often set by the government)

The control subsystem and the leasing algorithm, together, will make intersection collision efficient and safe.

2.6 Verification

To verify the properties of the system, we have designed 4 experiments. One for showing the efficiency of the algorithm, and two for showing the safety of the algorithm.

The first one is a comparison experiment, the experiment setup contains two V2V vehicles trying to pass the intersection at the same time from different directions and compare the total time to both cars to cross the intersection under our lease-based scheduling algorithm and the lock-based algorithm. We show that our algorithm is consistently 30% faster.

The second one consists of two V2V vehicles trying to cross the intersection at the same time. We show that the lease-based scheduling system is able to let both cars cross without crashing into each other.

The third one consists two V2V vehicles and one non-V2V vehicles trying to cross the intersection at the same time. We show that without V2V communication, the non-V2V vehicle will crash into the other V2V vehicle as due to visual obstacles, the two cars cannot

see each other. However, with our algorithm and communication, the V2V vehicle is able to slow down to avoid collisions with the non-V2V vehicle.

The fourth one consists of two V2V vehicles trying to cross the intersection at the same time. But before they cross, a sudden obstacle arise that blocked their ways. We show that the vehicles are able to do emergency stops and are able to recover from expired leases right after the obstacle is cleared out of their way.

During our tests of the system, we find that the above four experiments has a 100% success rate.

2.6.1 Engineering Feasibility and Future Improvements

In this project, we have shown that the "lease" concept can have great potential in advanced intersection traffic scheduling with a minimum working example. Simple as it is, we want to show that the "lease" concept actually enables further space utilization optimization. For example, we can split the intersection into multiple blocks which have independent lease management systems, to increase the space utilization.

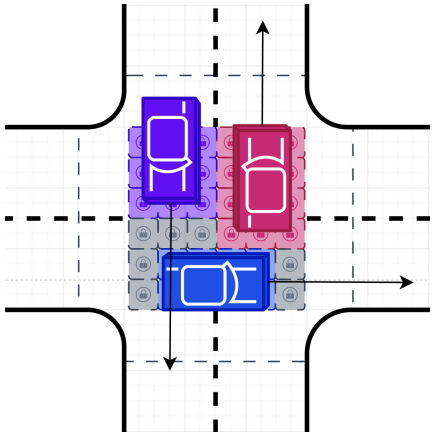

Figure 20: Lease Algorithm, Improved

Since lease application and the collision avoidance algorithm is largely limited by the accuracy of the prediction algorithm and the movement planning algorithm, the engineering team can easily, based on their specific needs, swap the existing prediction and movement planning algorithms with better ones or simpler ones to balance between performance and amount of computing resource available.

3 Costs

According to [9], the minimum hourly wage in Illinois is 8.3$. We set the desired hourly salary as 9$ based on that. During the first eight weeks, each partner contributes about 10 hours a week to the project, including writing documents, conducting experiment, and meeting with TA or professors. We take a break in the Labor Day Holiday and spend approximately 20 hours to refine VVCCS and prepare the final demo in the last week. In total, the labor cost for each member amounts to 225$, derived from the given formula:

In addition, we purchase a remote control car that costs 489¥ for the final demo.

4 Conclusions

4.1 Accomplishment

In summary, VVCCS achieves successful collision avoidance in all our experiments at intersection. The overall energy consumption is much lower than that required by traditional traffic control mechanisms, such as traffic lights and stop signs.

In this project, We implement the precise control of the speed of Qcar by combining the feed-forward and feedback term in the command to drive the motor, and realize the localization by converting the motor state to real speed and conducting the integration. The vision-based object detection subsystem achieves detecting other vehicles and pedestrians in the complex environment. Moreover, We exploit the etcd distributed data base to make vehicles communicate with each other. Lastly, We propose the idea of lease to help to schedule the timeline for each to pass through the intersection.

4.2 Ethical Considerations

Ethics is a crucial aspect of any new technology, especially those involving autonomous or semi-autonomous systems. Below are some key points we take into consideration:

Bias and Discrimination. We ensure that the algorithms and systems are designed to be fair and equitable, and that they do not perpetuate or exacerbate existing inequalities. When deciding forcing which vehicle to decelerate during a conflict, we adhere to the FIFO principle and give no special priority to any vehicles. Conduct Clause 1.4 Be fair and take action not to discriminate. [10]

Privacy. The use of V2V technology raises concerns about privacy, as it involves the exchange of sensitive information between vehicles. We ensure that personal information is properly protected and that the data is only used for the intended purpose. We also guarantee that vehicles cannot read and write data to other's private field in the distributed data base to prevent information leakage. Conduct Clause 1.6 Respect privacy.[10]

4.3 Limitation and Future Work

Although VVCCS achieves the safety and efficiency in collision avoidance and scheduling at the intersection, it has several limitations as well.

As we use the yolo5 model to recognize surrounding objects, we only detect the type of object in the perception module. We preset the size of each type of objects, and hard-code it into the system. However, even the same type of object, like the trucks, may have different sizes in the real world. It's necessary to upgrade the perception subsystem so that we can collect more comprehensive information. Possible solutions include merging the depth camera or echo location system into current one.

On the other hand, the VVCCS currently can only control the vehicle running in a line. It can nether control the vehicle to turn at the intersection, nor make the vehicle detour

when the blocking object remains stationary. To improve the system, we need better control and pose prediction module so that the vehicle can accurately acquire and release the lease and perform more intelligently.

If we can further develop VVCCS to eliminate the limitation, it will support more scenarios and have a wide employment. Our system is economically efficient and will significantly save the fuel cost, hence reducing the environment contamination. The V2V-based system will also save government's expenditure on the built on infrastructures that needed in V2X solutions. Therefore, VVCCS is prospective and deserves more technical and economical attention.

References

[1] D. Lord, I. van Schalkwyk, S. Chrysler, and L. Staplin, "A strategy to reduce older driver injuries at intersections using more accommodating roundabout design practices," *Accident Analysis & Prevention*, vol. 39, no. 3, pp. 427–432, 2007, ISSN: 0001-4575. DOI: https://doi.org/10.1016/j.aap.2006.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457506001710.

[2] R. Tay and S. Rifaat, "Factors contributing to the severity of intersection crashes," *Journal of Advanced Transportation*, vol. 41, pp. 245–265, Jun. 2007. DOI: 10.1002/atr.5670410303.

[3] V. Miller, *The impact of stopping on fuel consumption*, http://large.stanford.edu/courses/2011/ph240/miller1/, 2011.

[4] *Qcar*, https://www.quanser.com/products/qcar/.

[5] *Ros 2 document*, https://docs.ros.org/en/foxy/index.html, 2023.

[6] etcd Authors, *Etcd*, https://etcd.io/, 2023.

[7] L. Li and F.-Y. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 6, pp. 1712–1724, 2006. DOI: 10.1109/TVT.2006.878730.

[8] L. Li and F. Y. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 6, pp. 1712–1724, 2006.

[9] U. Institute, *Working during college*, https://collegeaffordability.urban.org/covering-expenses/working-during-college/, 2017.

[10] ACM, *Acm code of ethics and professional conduct*, https://www.acm.org/code-of-ethics, 2018.

Appendix A Qcar Dimensions

The dimension of Qcar is shown in Figure 21 and Figure 22.

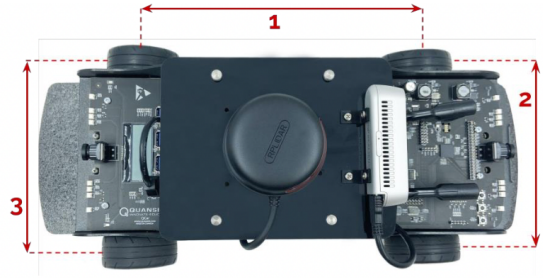

Figure 21: Qcar Dimensions

Item	Value
weight	2.7kg
Length	0.425 m
Height	0.182 m
Width	0.192 m
Tire diameter	0.066 m
Wheelbase (Figure 21 #1)	0.256 m
[l]Front and Rear Track (Figure 21 #2, 3)	0.170 m
Maximum steering angle	±30°

Figure 22: Dimensions