CISC322

Assignment 1
Conceptual architecture of Apollo
2022/2/14

Group: Steam

Kun Huang: 19kh2@queensu.ca

Haiyu Li: 17hl111@queensu.ca

Zhibing Liu: 18zl142@queensu.ca

Mingjia Mao: 17mm78@queensu.ca

Shunhao Tang: 18st13@queensu.ca

Jiajun Yang: 18jy56@queenus.ca

Table of content

CISC322 Conceptual Architecture

External Interfaces

Data Dictionary

Naming Convention

Conclusion

Lesson Learned

Reference

Abstract

Apollo is an open-source platform facing automobile and autonomous driving industries, established by Baidu. The Apollo platform is divided into four sub platforms, each serving a different purpose. In this report, we will discuss and focus on the conceptual architecture of its open software platform, which can also be divided into 6 major subsystems. We determined the High-Level architectural style for Apollo's open software platform is **Peer-to-Peer**. We will also go into depth of each subsystem in the software platform, explaining their functionality, dependencies and how they interact with each other. Besides, we will give an introduction on its concurrency model, explaining how the underlying system coordinates the processes synchronously. Moreover, two use cases with detailed explanations and sequence diagrams will show how the system reacts when facing different situations.

Introduction and Overview

Autonomous driving is becoming more developed as Artificial Intelligence is becoming a trending topic in the 21 centuries. Autonomous driving system allows drivers to free their hands and release their attention from the crowded traffic. More importantly, a well-developed autonomous driving system can provide prevention of severe traffic accidents from carelessness, drowsy driving.

This time, we are presenting our research on an Autonomous Driving platform developed by Baidu. The system was released on April 19th, 2017, named

after Apollo, the spaceflight program, aiming to build an open, robust platform for automobile industries to develop their own Autonomous Driving System based on this platform. The platform includes four different sub-platforms, Cloud Service Platform, Hardware Development Platform, Open Vehicle Certificate Platform, and Open Software Platform, which plays the core role within the system and will be detailed discussed in this report.

For the Open Software Platform, it provides the most essential technical components to achieve self-driving. They are Map Engine, Localization, Perception, Prediction, Planning, and Control units that work interactively as modules in a Peer-to-peer structure. The modules are working together based on CyberRT/ROS, an open-source operating system for robotics development as synchronous processes. We will also do a brief on the concurrency model, and how the model is optimized. For use cases, we came up with two scenarios, one of them is when the car meets obstacles and the other one is when the car meets Traffic lights. These scenarios are rather relevant with our everyday driving experience, thus the sequence diagram and explanation of the procedure of how the system deals with such scenarios provides a straightforward aspect on how data and commands flow as the system operates. We will also introduce the external interface, state how the software platform interacts with other components from a larger scale.

Lastly, we will provide a summary of our key findings and what directions we will be focusing on in the future.

Derivation Process

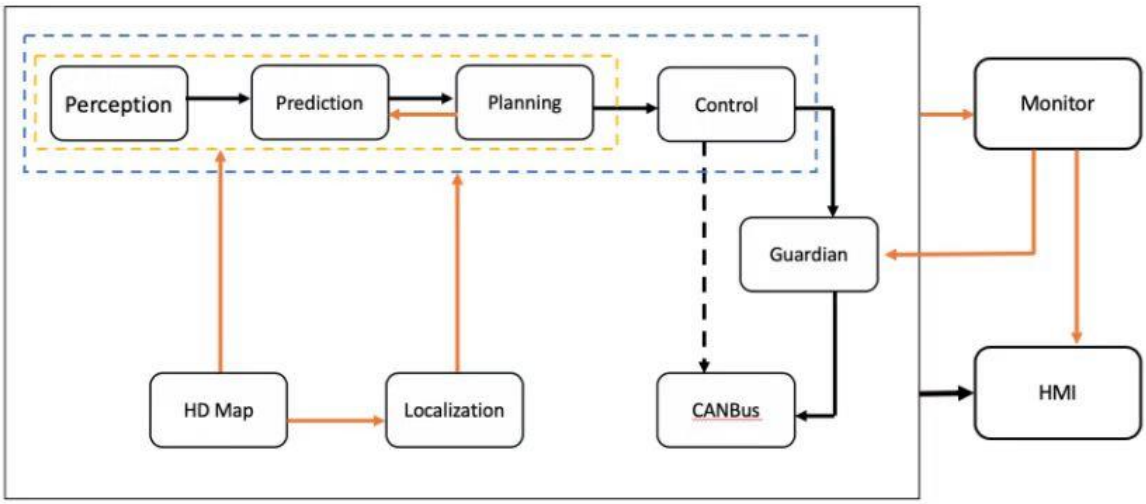
Determining the framework of the apollo software system requires a lot of research, collecting different information, including the introduction of the components, the principle of working together, the source code of the open platform, and some discussions in the forum. We need to clarify the original intention of Apollo's design and the content of the version update, including Apollo's original architecture design concept, because this is the foundation of a system, and subsequent updates are based on the original architecture. At the same time, in a large number of studies, we choose the content that is roughly the same as our idea for deeper discussion and needs to include the most important and authoritative citations to support the core arguments of our group.

Each member of our team needs to search the Internet for information and choose the structure that they think is the most suitable for apollo. When the group meeting is in progress, talk about the strengths and weaknesses of each, and combine the ideas of others to choose the most suitable structure. What matters is the different subsystems and the architectural style that binds them together. In the end, we settled on the seven most important components for apollo: map engine, localization, perception, pred

iction, planning, control, and HMI. We looked at the functional documentation for each subsystem and understood how they interact with the other components interact

Initially, we thought that a special module of the car was responsible for all the tasks and issued instructions to the subsystems. After further research, we found that this is not the case. The interaction of the subsystems is mutual, and many components will transmit information to each other. There is no such thing as a single component that does all the work. The map engine and localization are functionally similar components, and the core components of the system can be perception, planning, and control. These three components roughly confirm the working principle of the system, detect different situations according to the radar and control the vehicle to operate correctly through the planning module and the control module

Conceptual Architecture



This architecture is a mix of both peer-to-peer architecture and Publish/Subscribe architecture, allowing us a commonly used computer networking architecture in which each workstation, or node, has the same capabilities and responsibilities. It also serves as a distributed computing system where multiple processes are split between servers.

Evolution of System

The overall structure of the Open Software Platform has not changed much, according to reports from the ten versions of Apollo. Most of the updates are subdivision and expansion of functions. The first-generation Apollo is mainly composed of Localization, planning and control. Map Engine and Perception were added in the second version, and the component prediction was added in the sixth version. Until the tenth version, V7, no new component was added. Before and after these releases are mostly improvements in hardwa

re, security, and algorithms. Improve the categories of commands the car can execute, such as obstacle and traffic light recognition, vehicle stability and perception of complex road changes

Subsystems

Map Engine

Map Engine plays an important role and can be regarded as a fundamental component for the whole software system. It interacts with almost every module in the system. Map Engine contains an enormous amount of driving-assistance information.

When users are driving, and they open navigation software, the system always gives several optional routes, and for most cases it provides traffic information such as traffic jams, speed limit or traffic control. That information is sufficient for a human being to drive but not an Autonomous Driving Agent, since drivers can locate themselves by sensing the surroundings and the location GPS provided, drivers can also sense the position of traffic lights, width of roads rather easily. However, for Autonomous Driving Agents, a High-Precision Map is needed to store all the information introduced above to support the system.

Usually, GPS/ Navigation Software on mobile phones provide meter-level precision maps, and for modern Autonomous Driving Agents, the precision usually needs to achieve centimeter-level or even better, millimeter level.

The Map Engine plays the core of the whole system. It interacts with localization, perception, and planning modules. For localization modules. The map engine first gets the approximated longitude and latitude from GPS and then, based on cameras, radars built on the car, the map engine computes the accurate location of the vehicle. For perception, when perception modules meet difficulties sensing potential obstacles. For example, during bad weather, the perception unit is limited, it might not sense the traffic light due to poor sight, the map engine now provides extra information, in this case, an incoming traffic light to help the agent make better decisions. For the planning module, it cooperates with the map engine to help the driving agent plan better routines, and also help predict other cars' potential track.

The core technology for Apollo is its Map Engine, Baidu has its own map database on Apollo Cloud Service Platform and can interact with Map Engine.

CISC322 Conceptual Architecture

Perception

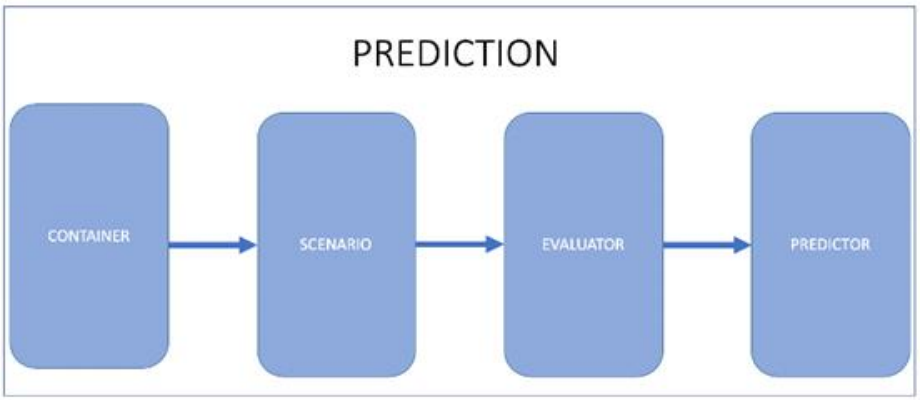
Perception is one of the most important subsystems in autonomous driving. Basically, perception has two main components which are sensing component and perception components and each of those components has their own sub-component as well. There are also other components that are important as well.

Sensing means gathering data which are physical variables with different sensors. In Apollo v7, those sensors are mostly the Camera component, Lidar Component and Radar Component. The sensing component should take photo, lidar and radar from outside as input and generate useful information as out and transform that information to perception component.

Perception refers to the semantics of that data which can tell and understand what the real environment is and how to react based on reality. The perception component should take the information that is gathered by sensing component as input and generate two results as output: Final Traffic Lights and Final Objects with Type, Distance and Velocity. There are many sub-components inside perception sub-systems. First, image and point cloud pre-processing. These processing components are used to pre-process the information from sensing components. As we know, in an artificial neural network, the data needs to be pre-processed and then be used in the deep networks. Second, the system needs the deep network to process the data for recognition, tracking and conversion. After processing the data, a post-processing component is also required so the system can do the fusion work to generate the correct final outputs.

Perception component has a great relationship with localization. These two components work together and influence each other. There are two main improvements in Apollo v7 than other versions which are the improved camera-based Obstacle Detection Model and improved Lidar-based Obstacle detection model. SMOKE is a single-stage monocular 3D object detection model which made some improvements based on the CenterNet. The developers have done some adaptations on the SMOKE and trained on waymo open dataset. The improved Lidar-based Obstacle detection model is a new LiDAR-based obstacle detection model named Mask-Pillars based on Point Pillars, which improves the original version in two aspects. The first one is that a residual attention module is introduced into the encoder of the backbone to learn a mask and to enhance the feature map in a residual way. The second one is that a pillar-level supervision is applied after the decoder of the backbone which is only performed in the training stage.

Prediction



Prediction module predicts the future motion trajectories of perceive d obstacles detected by the perception module. It receives obstacle informa tion from the perception module and localization information from the local ization module and it will return the annotated obstacles with predicted tr ajectories and their priorities based on the previous computing cycle from the planning module. The module consists of 4 main functionalities: contain er, scenario, evaluator, and predictor. Container submodule stores input da ta from the perception, localization, and planning module. The data passes to the Scenario, Evaluator and Predictor sub-module to analyze the scenario including ego vehicle and all the obstacles perceived. Currently, Scenario sub-module defines scenarios as Cruise (includes lane keeping and followin g) or Junction (have traffic lights and/or stop signs). Evaluator separates obstacles into three priorities: Ignore, Caution and Normal based on the ef fect of the ego car’s trajectory. Obstacle priority in different scenarios is calculated differently. Predictor sub-module generates the predicted tra jectories for obstacles. The Prediction module finally outputs annotated ob stacles with predicted trajectories and priority.

Localization

The system of autonomous driving is divided into many parts. Many tec hnologies are involved, from sensor perception to subsequent recognition, l ocalization, decision making, and then to the control system. One of the co re autonomous driving technologies is high precision localization. This is because vehicles wanting to achieve autonomous driving on the road need the vehicle itself to be positioned with centimeter-level positioning accuracy, otherwise, accidents may occur that do not meet expectations. For example, if the positioning module is not accurate enough, it may not be able to dri ve on the correct path or collide with an obstacle. Meanwhile, positioning involves a lot of feature extraction, such as information about the surroun ding environment. Hence, the localization module and the perception module are combined. If the perception module is not done well, the accuracy of po

sitioning will also be reduced. Apollo localization method provides two different implementations for different application requirements, one is RTK positioning method which the Apollo system mainly uses and another one is MSF positioning method.

RTK positioning method is a GPS+IMU global positioning navigation system that can achieve centimeter-level positioning accuracy with high-quality GPS signals. The MSF positioning method fuses information from multiple sensors, including GPS, IMU, and LiDAR.

The localization module mainly implements the following two functions: the first one is to output the position information of the vehicle for the planning module, and the second one is to output the attitude and speed information of the vehicle for the control module. In the data processing of the localization module, for GPS data, the data will be copied directly to the output data. For processing IMU data, the system will first convert the IMU information relative to the earth's east/north/up coordinate system into IMU information relative to the vehicle's own x/y/z coordinate system, and then the data will be copied to the output data. LIDAR is used to assist in the case of poor GPS signals. It also obtains real-time environmental information from the sensors and compares it with recorded static reference information like road signs to calculate the vehicle's position.

Planning

The function of the planning module is to plan a trajectory that the vehicle can travel based on the results of the perception prediction, the current vehicle information and road conditions, and this trajectory will be handed over to the control module. The control module makes the vehicle through the accelerator, brake and steering wheel Follow the planned trajectory.

The trajectory of the planning module is the short-term trajectory, that is, the trajectory of the vehicle in a short period of time, and the long-term trajectory is the navigation trajectory planned by the routing module, that is, the trajectory from the starting point to the destination, followed by a short-term trajectory until the destination. This is also easy to understand. We turn on the navigation before driving, and then drive according to the navigation. If there is a car in front, we will slow down or change lanes, overtake, avoid pedestrians, etc. This is a short-term trajectory, combining the above methods until we reach the destination.Planning is a feature of most, Apollo 2.0 needs to use multiple sources of information

to plan safe and collision-free driving trajectories, so the planning module interacts with almost every other module.

First, the planning module obtains the output of the prediction module. The predicted output encapsulates the original perceived obstacle, and the planning module subscribes to the traffic light detection output instead of the perceived obstacle output. Then, the planning module obtains the routing output. In some cases, if the current routing result is not executable, the planning module can also trigger a new routing calculation by sending a routing request.Finally, the planning module needs to know the location information (location: where am I) and the current autonomous vehicle information. The planning module is triggered by a fixed frequency, and the main data interface is the OnTimer callback function that calls the RunOnce function.

Data dependencies such as chassis, positioning, traffic lights, and forecasts are managed through the AdapterManager class. The core software modules are also managed by the AdapterManager class. For example, localization is managed via Adapter Manager:: GetLocalization.

Control

The control module is one of the core modules of Baidu Apollo, which is used to receive data, generate instructions through algorithms, and pass them as output to the next component. There are three types of input, the first is the trajectory based on the planning module, the second is the current state of the vehicle, and the third is the vehicle position and speed information generated by localization. The control module calculates the car's accelerator, brake, and steering signals based on these three inputs, and drives the car according to the prescribed trajectory for control. The control method is based on vehicle dynamics and kinematics knowledge to model the vehicle.

Apollo V7 doesn't have much of an update to the Control component, as it relies heavily on algorithm evolution and improvement. The most recent update V5.5 release added the new MRAC algorithm as well as the Control Profiling Service. This algorithm can compensate for dynamic delay and time latency of steering-by-wire effectively. It also ensures faster, more accurate steering control actions to track planned trajectories.

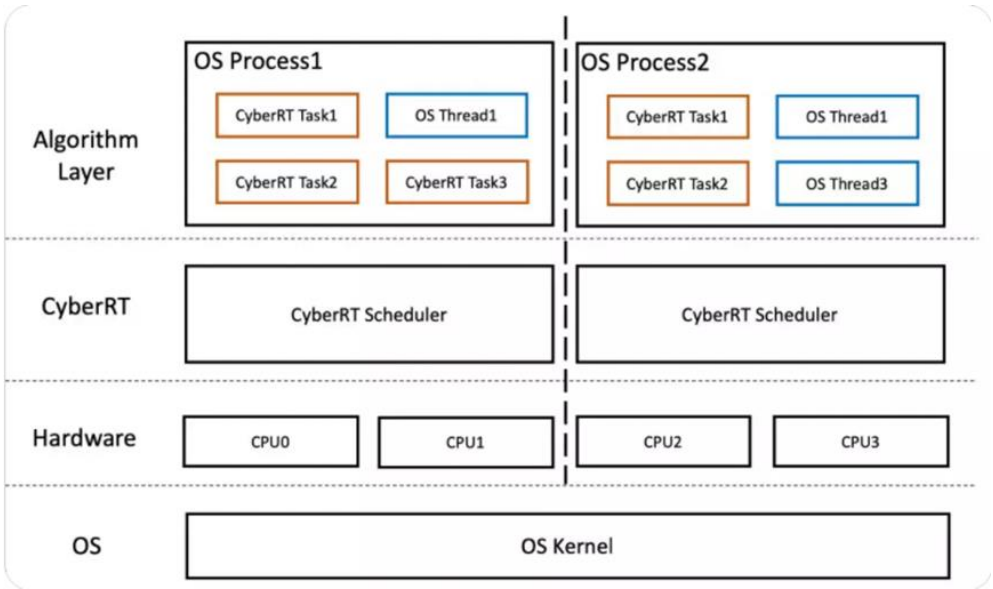
Cocurrency Model

In the autonomous vehicle operating environment, in order to improve the concurrency performance, the algorithm module will create a large number of asynchronous tasks. Traditional asynchronous tasks problems are implemented by OS Thread and dispatched by the OS Kernel which will lead to some

limits. Using such mode will have some limitations. During the process, users cannot control the core sequence and execution periodicity of the tasks. As a result, the real-time performance of autonomous driving computing tasks cannot be guaranteed. As the OS Kernel uses the time slicing algorithm to dispatch these asynchronous tasks. When the number of asynchronous tasks increases, the cost on Context Switch is also increased. Also, the cost on Cache Miss and Cache Bouncing will increase if there is a large number of irrelevant asynchronous tasks using CPU resources at the same time.

In the Apollo software platform, the team used Apollo Cyber RT to calculate the task dispatch and provide a communication service for low-level components. It solved the problems of the previous ROS framework and is more suitable for autonomous driving.

In the Cyber RT framework, the scheduler transfers algorithmic asynchronous tasks from OS Thread to Cyber RT Task in user space. Meanwhile, OS Kernel CPU scheduling policy configuration interface is encapsulated to control OS Thread and OS Process in the process of dispatching CPU resources in OS Kernel.



The algorithm layers have three kinds of asynchronous tasks: OS Process, OS Thread and Cyber RT Task. Cyber RT Scheduler is between algorithm layer and OS hardware to schedule the CPU resources in Cyber RT Task. Because most of the asynchronous tasks are transferred from OS Thread to Cyber RT Task, the Context Switch and the order of Cyber RT Task is scheduled by Cyber RT scheduler. OS Kernel is not required to evolve in the process. The rea

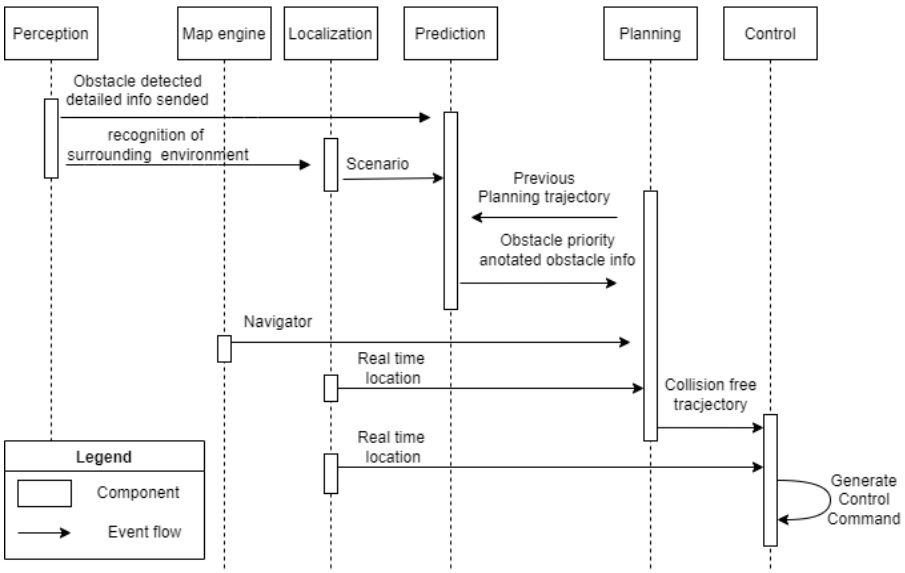
l-time and periodicity of asynchronous tasks at the algorithm layer is guar anteed. Also, due to the number of OS Thread decreases, the cost on Context Switch is also reduced. Different OS Processes can be separated in CPU reso urces by Cyber RT Scheduler, the cost of Cache Miss and Cache Bouncing is a lso reduced.

Use Cases:

Use cases 1: Obstacles

Assuming that the vehicle has become autonomous driving, when there i s an obstacle on the front road, how does the software system react and con nect various components to work together to avoid the obstacle

Figure Obstacle avoidence of automatic driving



1. Obstacles detected by Lidar and detailed info will be sent from Percepti on Module to Prediction Module.
2. Recognition of the surrounding environment detected by Lidar will be sen t to the Localization Module.
3. Localization Module sends localization info to Prediction Module to upda te scenario state.
4. Planning trajectory of the previous computing cycle from the Planning Mo dule to Prediction Module.
5. Prediction Module sends obstacle priority and annotated obstacle info to Planning Module.
6. Map Engine sends navigator info to the Planning module.
7. Localization Module sends real time location to Planning Module and Cont rol Module.

8. The Planning Module sends a collision-free and comfortable trajectory fo r the control module to execute.
9. Control module generates control commands to the chassis.

Use case 2: Traffic lights

1. Traffic detected by traffic light components and send the data to tempora l and multiple traffic light recognition components.
2. The recognition of the traffic light environment by the perception compon ent will be sent to the localization module.
3. Localization module sends the localization information to the prediction model’s container and activates the Junction Scenario.
4. Planning module receives feedback from Localization, perception and predi ction module and activates traffic light scenario handler.
5. Planning trajectory of the previous computing cycle from the Planning Mod ule to Prediction Module.
6. Map Engine sends navigator info to the Planning module.
7. Localization Module sends real time location to Planning Module and Contr ol Module.
8. The Planning Module sends a collision-free and comfortable trajectory for the control module to execute.
9. Control module generates control commands to the chassis.

External Interfaces:

The external interface records all information transferred to and from the system. This includes a graphical user interface, file management, databas e, message or network.

As we can see from the website of Apollo hardware, there are plenty of comp onents which construct the physical Apollo system. There are radar, camera, navigation, the Controller Area Network (CAN), Apollo Sensor Unit (ASU), Ap ollo Extension Unit (AXU) and Apollo Computing Unit. The base open platform supports these components.

Data Dictionary:

Map engine: Provides specific structured information about the road

Perception: The surrounding environment of the autonomous vehicle. There a re two important sub-modules in perception: obstacle detection and traffic light detection

Prediction: Make correct and appropriate judgments about what is likely to happen in the future

Localization: Using various sources of information from GPS, LiDAR and IMU to locate the position of autonomous vehicles

Planning: Plans the time and space trajectory of autonomous vehicles

Control: Using different control algorithms according to the planned trajectory and the current state of the vehicle, to generate command

HMI: Component for viewing vehicle status, testing other modules and controlling vehicle functions in real time

Subsystem: Component of the whole software system

Cyber RT: An open source, high-performance runtime framework designed for autonomous driving scenarios

Naming Convention:

RTK (Real Time Kinematic): The invention relates to a technology for improving GNSS positioning accuracy by using a fixed base station

IMU (Inertial Measurement Unit): Inertial Measurement Unit, mainly provides object attitude information, such as angular velocity and acceleration of three axes.

GPS (Global Positioning System): Global Positioning System, which mainly provides information on the longitude, latitude, and altitude of the object.

LiDAR (Light Detection And Ranging Sensor): Laser detection and measurement. The measured data is a discrete point representation of the Digital Surface Model (DSM), which contains spatial three-dimensional information and laser intensity information.

MSF (Multi-sensor Fusion): Provide a robust method which can achieve high localization accuracy and resilience in challenging scenes, such as urban downtown, highways, and tunnels

OS(operating system): Systems software which used to manage hardware, software, and provide assistance for a service

CPU(Central processing unit): principal part of any digital computer system

Conclusion:

In conclusion, through a lot of research and collaboration in our group, we consider that the Apollo project is characterized by the fact that Apollo is an open and secure platform, so people can readily understand the structu

re and concepts of this platform, which helps the automotive industry and the autonomous driving domain to combine and develop. As we explore this area and based on what we learned in the course, we believe that the architecture style of the Apollo open source software platform is pub-sub for high-level and peer-to-peer for low level. Since the Apollo system includes many different individual components, those components work together, then put all nodes in a common area, and each node stores information about the other nodes, when a node has a problem it does not affect the operation of the whole system. This enables people to experience the Apollo system in a more secure and stable way.

For future development suggestions, we consider that self-driving technology needs to be more accurate and safe, as we still sometimes see news about people causing accidents due to self-driving cars. Although self-driving technology brings us a lot of benefits, humans-driven vehicles cannot be completely replaced while the technology is still not fully mature and safe.

Lession Learned

The Apollo system consists of six modules, so each person was assigned to understand the assigned module and write down what they know. However, this does not mean that we only focus on our own module, because each module is related and then forms a whole. Therefore, we not only need to have an understanding of our own module, but also need to understand other modules that can better help us understand the whole concept of the system.

Reference

ApolloAuto. (n.d.). *ApolloAuto/apollo: An open autonomous driving platform*. GitHub. Retrieved February 19, 2022, from https://github.com/ApolloAuto/apollo

Ding, B. (2020, October 14). *Overview of Apollo software system*. Retrieved February 19, 2022, from https://dingfen.github.io/apollo/2020/10/14/apollo-intro.html

ApolloAuto Developer Centre (n.d.) *CPU resource scheduling and data and co routine task-driven model of Cyber RT in autonomous driving system* https://apollo.auto/developer/index_cn.html#/learning?id=7

Li, K. (n.d.). *Baidu apollo3.0 software architecture details*. Zhihu. Retrieved February 19, 2022, from https://zhuanlan.zhihu.com/p/48654886

Hong, D. K., Cao, Y., Jin, Y., & Kloosterman, J. (n.d.). *AVGuardian: Detecting and mitigating Publish-Subscribe Overprivilege for Autonomous Vehicle Systems*. Retrieved February 19, 2022, from https://www.ics.uci.edu/~alfchen/david_eurosp20.pdf

He, Bote. *Baidu Apollo Decision Making Modules and Architecture.*

https://blog.csdn.net/weixin_43795921/article/details/105730561

Ding, Bill. “Apollo Software System Intro.” *FengZiLeYuan*, Retrieved 14 Oct. 2020,

dingfen.github.io/apollo/2020/10/14/apollo-intro.html

CISC322 Conceptual Architecture