Parking Robot based on 3D LiDAR

Author: Hu, Yadong,   Supervisor: Dr. Sun, Yongkui
SWJTU-Leeds Joint School, Southwest Jiaotong University

Abstract

Motivation:
- Driving in parking lots is dangerous and time-wasting.
- It's hard to balance the precision (function) and price of 3D LiDAR.
- No remote control and visualization system.

Work:
- SLAM+: Based on Cartographer but adding IMU factors. More precise mapping results can be obtained from a cheap 3D LiDAR.
- Optimized RRT real-time dynamic 3D navigation: Using the 2D OGM map + height information obtained in real-time from sensors.
- Scalable and portable Client-Server-Vehicle (ROS) architecture with VLP-16 3D LiDAR simulation; Model maps of an outdoor and an indoor scenario; communication middleware; Android control App.

Result:
- Real-time obstacles detection and dynamic path planning. The robot can safely pass speed bumps and arched (limited height) gates.
- Although the map building is 10% slower than built-in method, its 3D restoration is more precise and accurate, and the RRT spends an average of 23% less path calculating time in complex environments.
- Parking and calling the vehicle automatically and remotely.

Literature Review

Google Cartographer:

$$\text{odds}(p) = \frac{p}{1-p}$$

$$M_{\text{new}}(x) = \text{clamp}\left(\text{odds}^{-1}\left(\text{odds}\left(M_{\text{old}}(x)\right) \times \text{odds}\left(p_{\text{hit}}\right)\right)\right)$$

A square represents the map block, and the data stored inside indicates the probability of being occupied. Send a laser beam, and it hits an obstacle point called the hit point, and the empty area in the middle of the line is called the missing point. 2D and 3D information are stored in such a map.

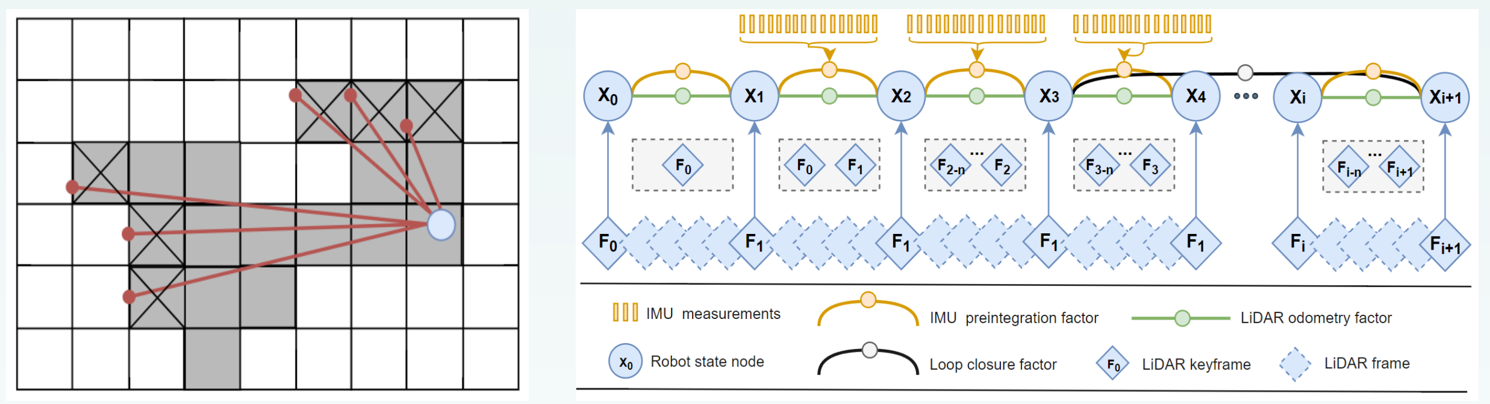


Fig. 2-1: Left: Theory of Cartographer. Right: Framework of LIO-SAM.

LIO-SAM:

LIO-SAM is the sequel of LeGO-LOAM with the IMU pre-integration factor and GPS factor. This framework uses Keyframe for matching, dropping frames between keyframes.

Table 2-1: Comparison of modern SLAM methods.

	Cartographer	LOAM	LIO-SAM
Speed	Very Fast	Low $O(n^6)$	Fast
Precision	Standard	High	Very High
Features	Complete interface, Toward industry	Time stamp, Scan-to-scan and Map-to-map	IMU pre-integration and GPS factor
Drawbacks	Fully encapsulated, not easily modified; Low accuracy in building maps	Need to extract plane and edge points; Cannot handle large scale transformations	IMU not easy to be configured

Improvement

1. SLAM+

Based on the Cartographer developed by Google, the project refers to introducing the IMU factor proposed to significantly optimize positioning effect to improve the accuracy of map building. This is easy to implement in Gazebo simulation, just pass the IMU data to the open-source algorithm and then pass the converted and processed PCD data to RViz for display.

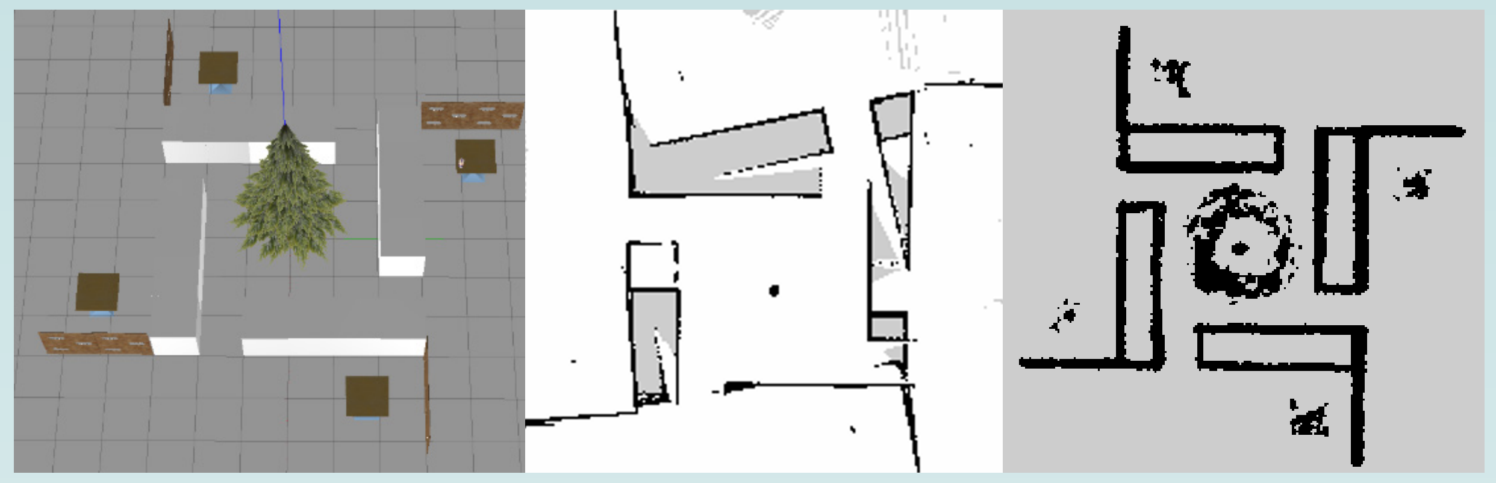


Fig. 3-1: Comparison of tree mappings. Left: Real scene. Middle: Cartographer 2D result. Right: SLAM+ result.

2. Optimized RRT 3D Dynamic Navigation:

2D OGM maps and real-time height information can significantly optimize the drawbacks due to the determined map. Usually, the fixed obstacles are already recorded in the OGM map in advance, so it is only necessary to detect whether there are dynamic objects or static obstacles below the threshold on the planned path in real-time to achieve dynamic navigation.

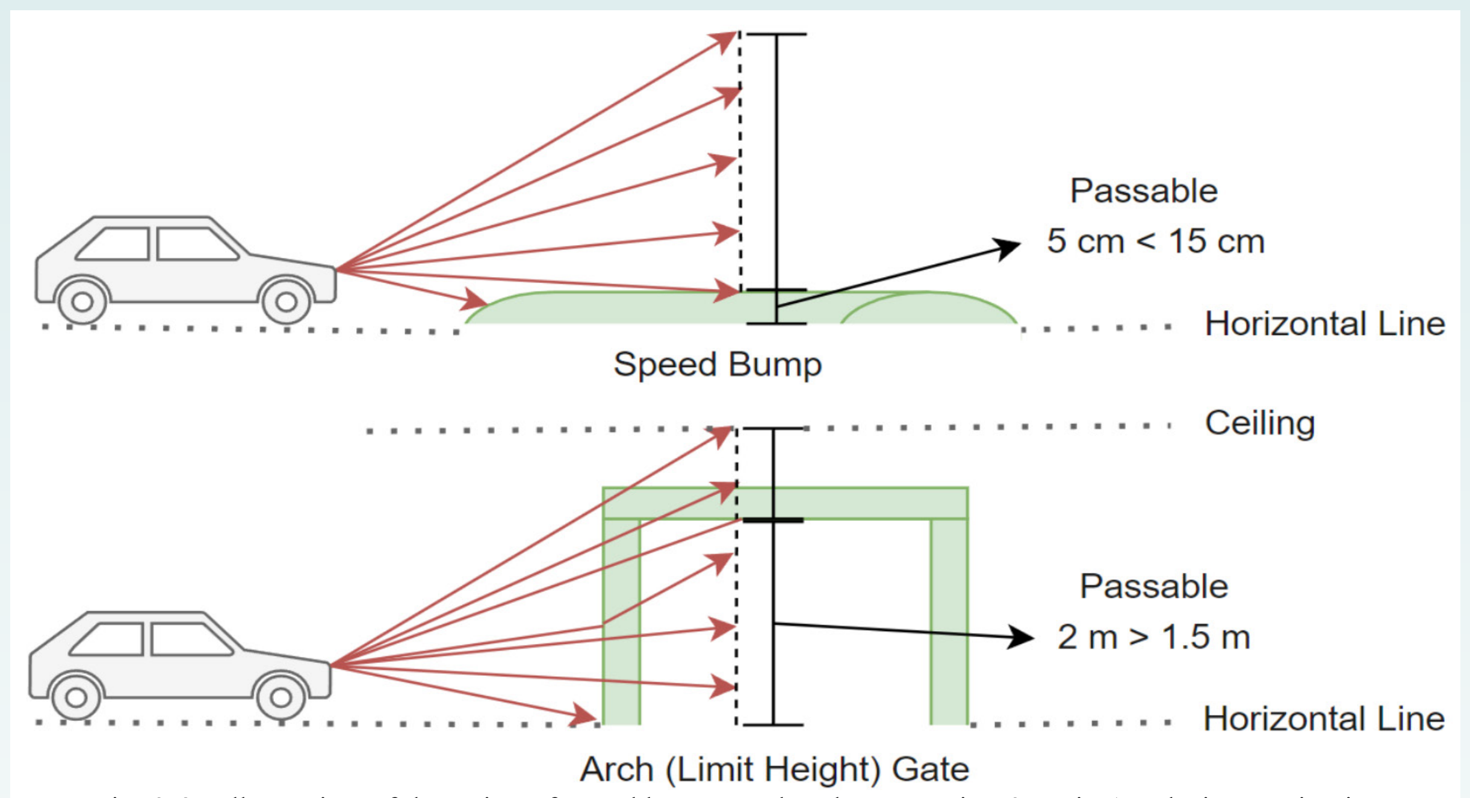


Fig. 3-2: Illustration of detection of speed bumps and arch gates using 3D LiDAR during navigation.

Results

1. Client-Server-Vehicle (ROS) Architecture

The client part represents the user and the smartphone application, through which the data is sent and received. The server part is included in the virtual machine. An Ubuntu server written in Kotlin language acts as an information transfer station, all commands are encoded and decoded by this server. The vehicle part in this simulation project is the ROS virtual system, where Gazebo software simulates the 3D LiDAR and the 3D model, RViz renders the Octomap for visualization.

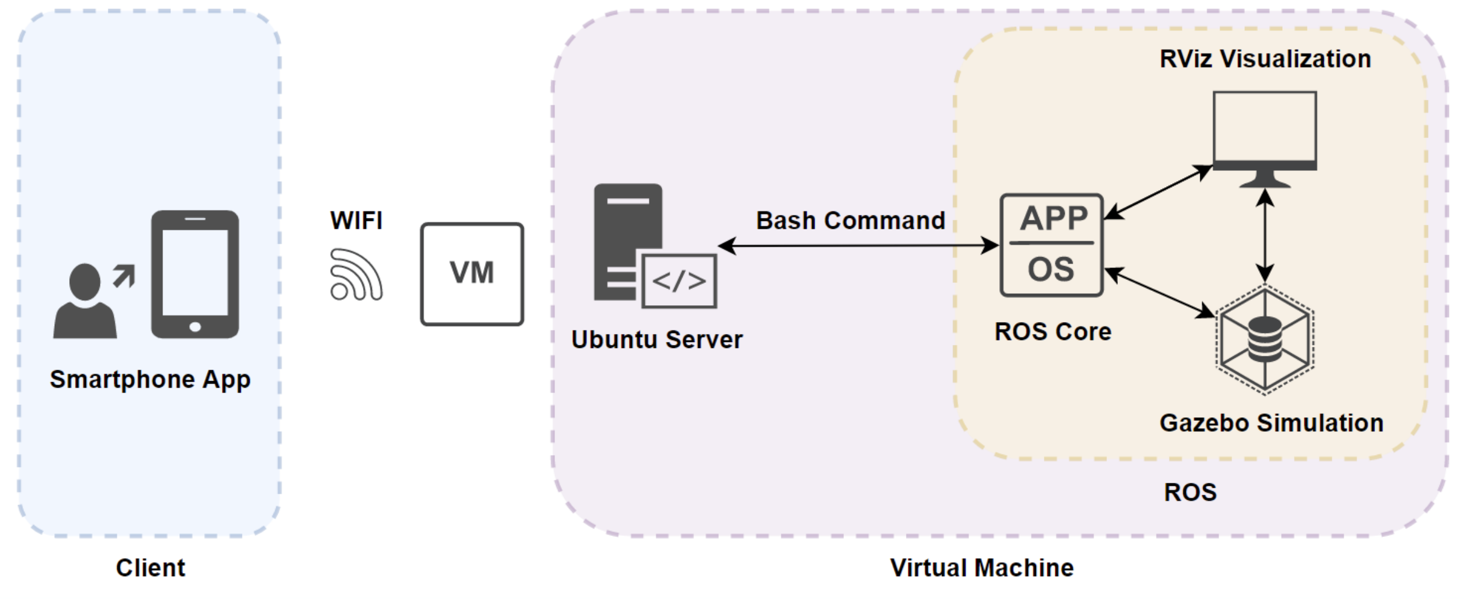
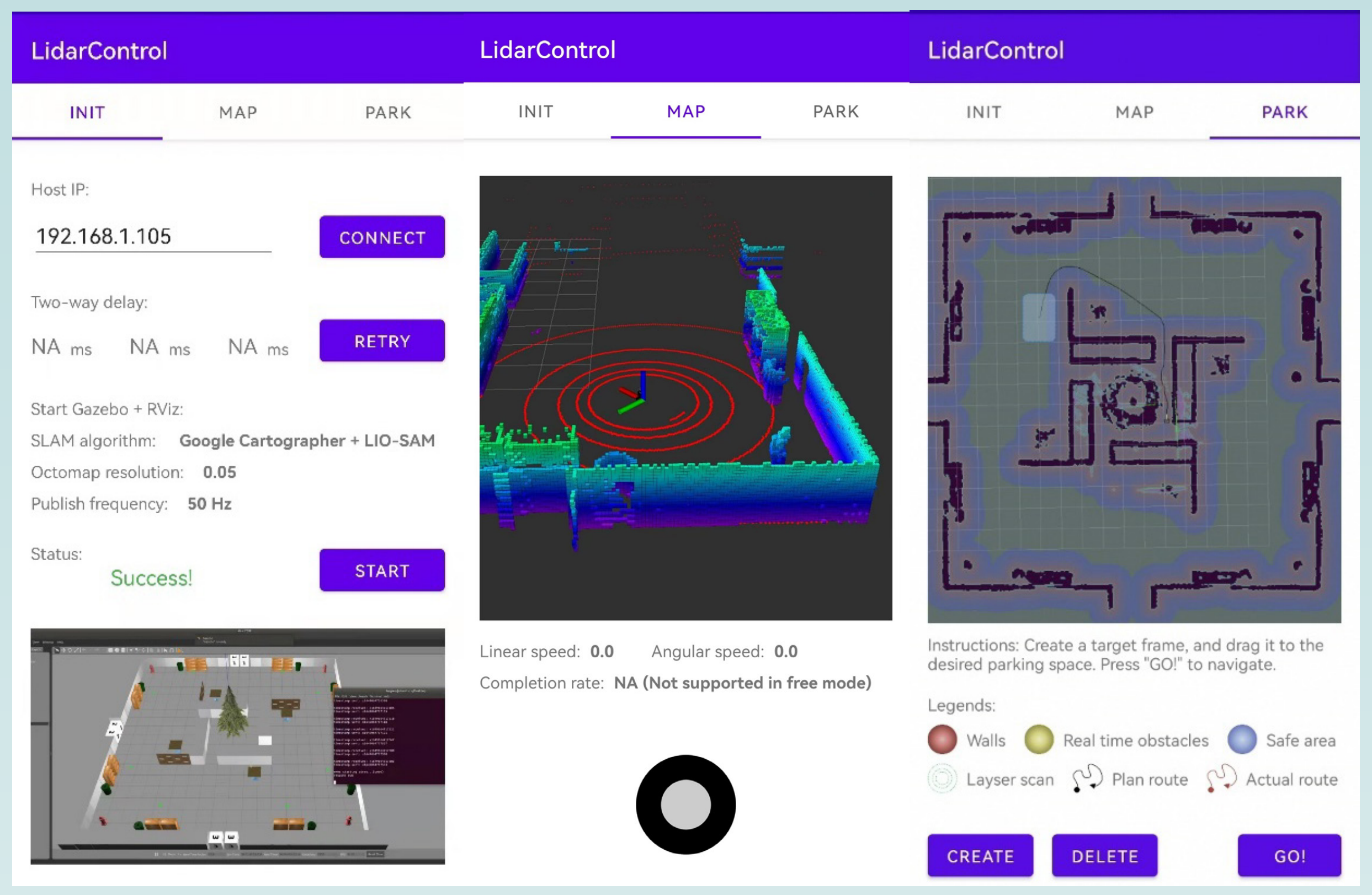


Fig. 4-1: The Client-Server-Vehicle architecture. VM stands for Virtual Machine. The server part and the car part are both included in the VM. In this simulation, the vehicle layer is embedded in the ROS.

Results



Host IP: 192.168.1.105
Two-way delay: NA ms NA ms NA ms
Start Gazebo + RViz:
SLAM algorithm: Google Cartographer + LIO-SAM
Octomap resolution: 0.05
Publish frequency: 50 Hz
Status: Success!

Linear speed: 0.0 Angular speed: 0.0
Completion rate: NA (Not supported in free mode)

Instructions: Create a target frame, and drag it to the desired parking space. Press "GO!" to navigate.
Legends: Walls, Real time obstacles, Safe area, Layser scan, Plan route, Actual route

Fig. 4-2: Android control and visualization application. Left: Initial. Middle: Mapping. Right: Parking.

2. Scenario 1: Outdoor Parking

Difficulties: A large tree in the middle of scene and a narrow path around it. Common obstacles: tables, fenders, fire hydrants, trash cans. This scenario tests the vehicle's recognition of small objects and ability to pass through the dense tree areas correctly.

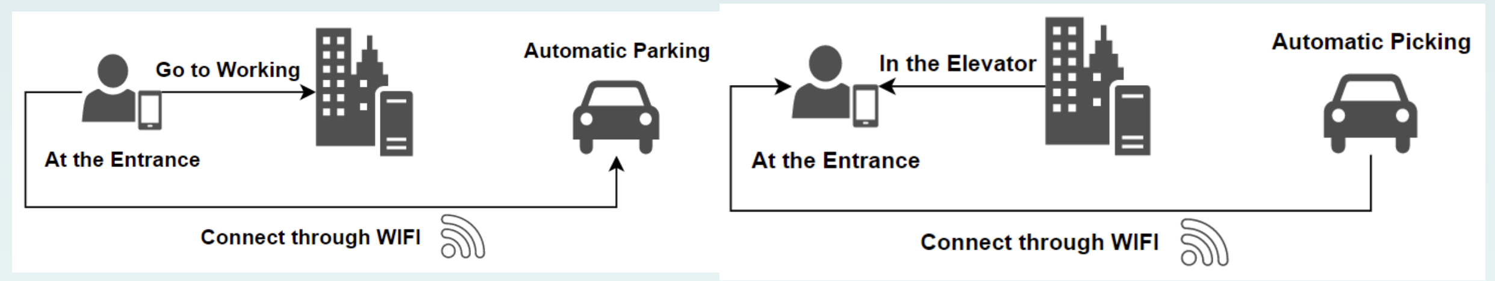


Fig. 4-3: Two life scenarios. Left: scenario 1, people get out of their vehicles at the entrance. Right: scenario 2, the vehicle picks people at the entrance after work.

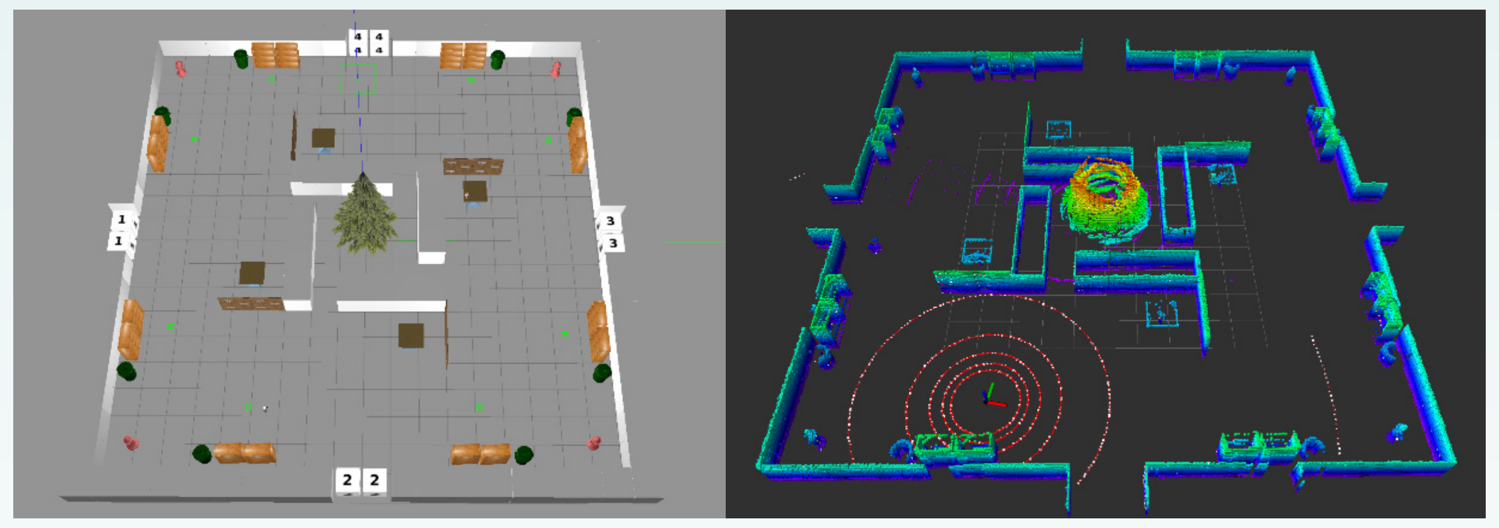


Fig. 4-4: Map building result. Left: Real scene. Right: This system's 3D restoration result.

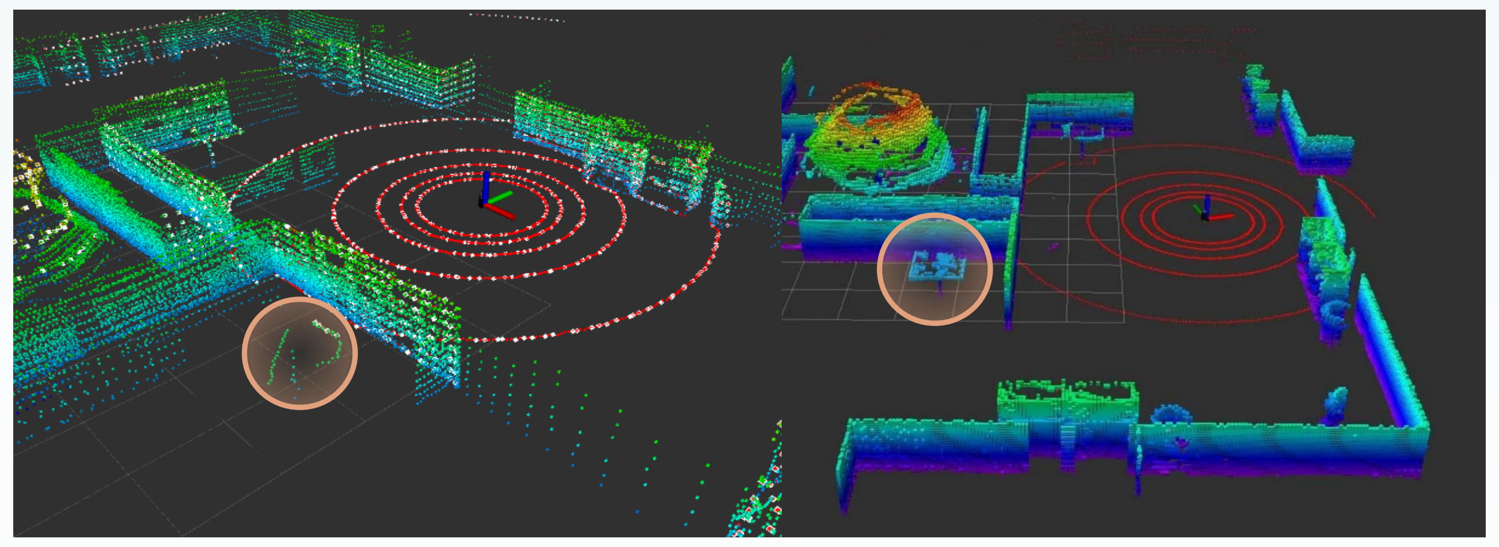


Fig. 4-5: Comparison of the table (detail) between Cartographer (Left) and SLAM+ (Right).

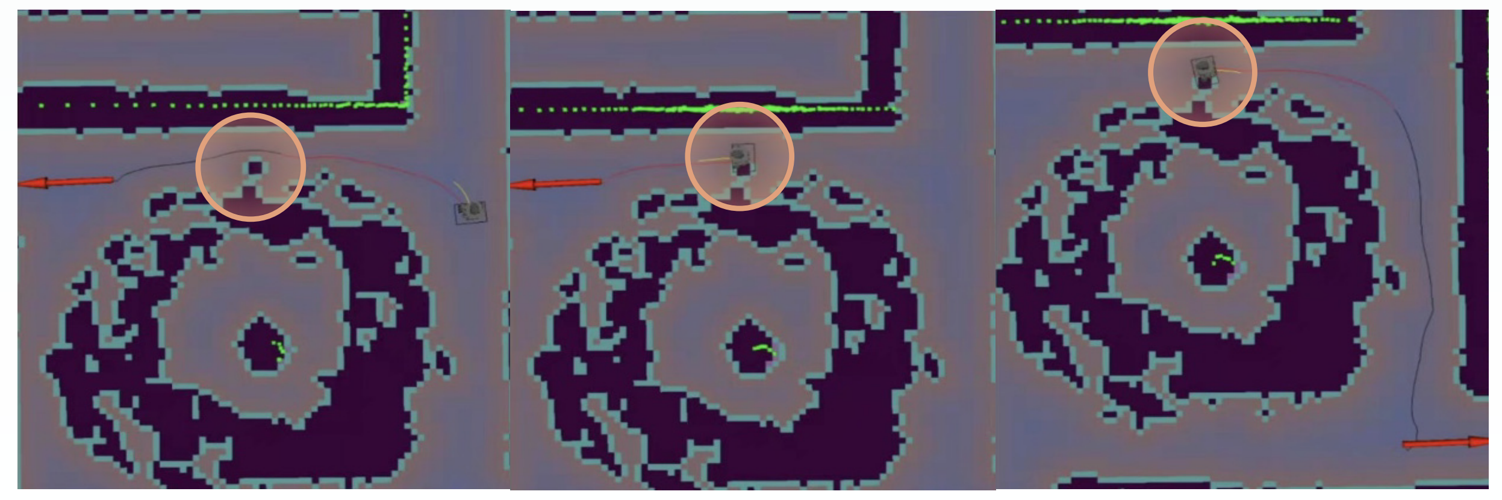


Fig. 4-6: Real-time path planning example. Left: Initial path bypasses the obstacle. Middle and Right: When the vehicle approaches the obstacle, it finds that its height is safe to pass, so it changes for a nearer path.

Results

3. Scenario 2: Indoor Parking

Difficulties: Parking spaces with size restrictions, speed bumps simulated by steps, and numerous suspended objects simulated by arched structures. Vehicle should smoothly recognize and pass through the speed bumps and height restriction gates.

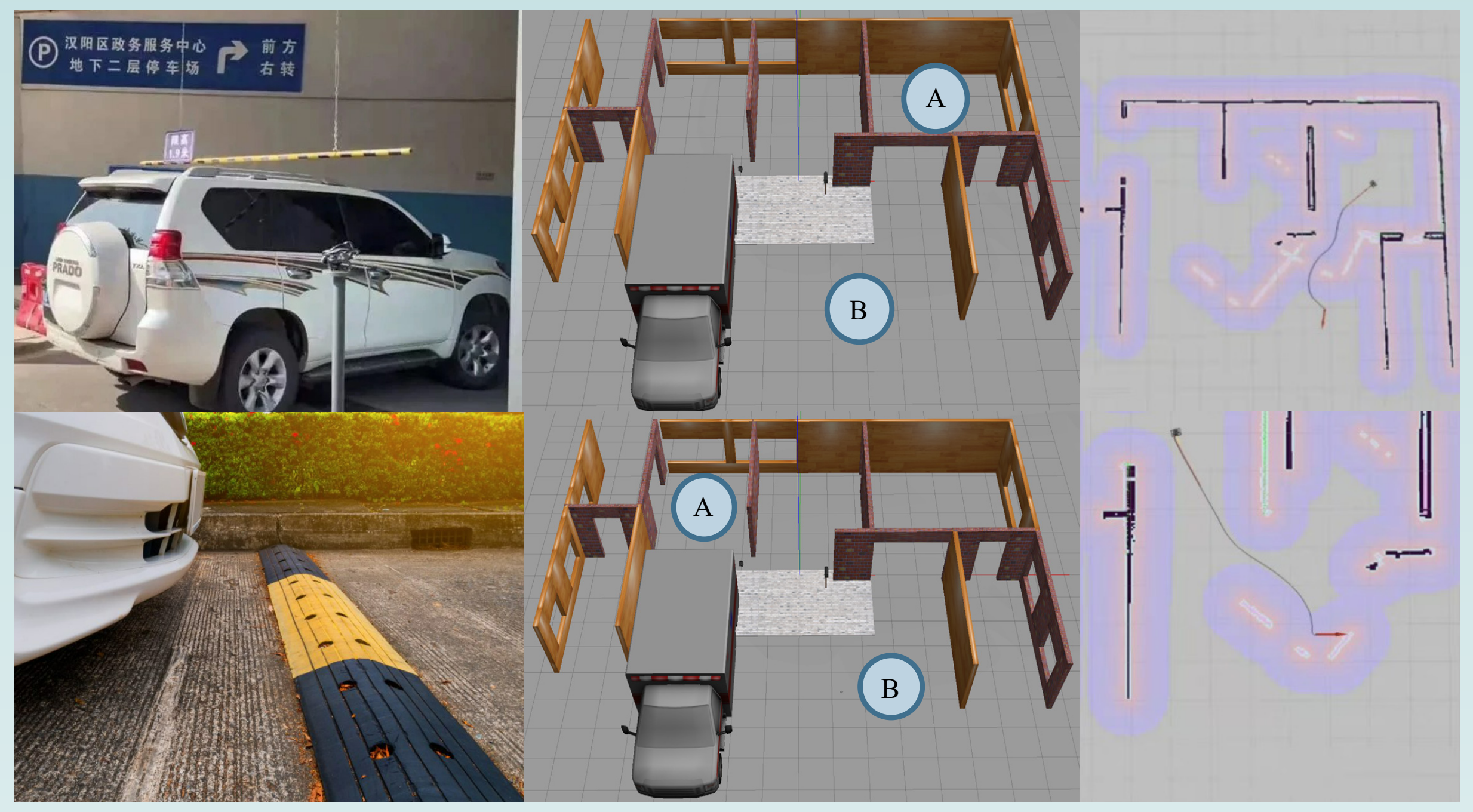


Fig. 4-7: Speed bumps and arch gates passable test. Top: From point A to B, it must pass the arched gate for shorter path. Bottom: It must pass the speed bump for shorter path.

Conclusion and Future Work

This system improves SLAM algorithm concerning the stability brought by the addition of the IMU factor so that errors from multiple scans during map building no longer accumulate. Therefore, the obtained model is more refined and accurate, but the build time is slightly longer.

Table 5-1: Running time of mapping at a completion rate of 90%.

Algorithm	Scenario 1: Outdoor (s)	Scenario 2: Indoor (s)
Cartographer	487	239
SLAM+	566	258

The paths planned by RRT are related to the sampled points, not smooth, and may have sharp turns, and the robot cannot strictly follow the paths when it runs, and the increase of turns also makes the robot running time increase accordingly, especially in simple paths.

Table 5-2: Comparison of the performance of popular navigation algorithms.

Algorithm	Simple Path Navi. Time	Complex Path Navi. Time	Far Dest. Navi. Time	Far Dest. Calc. Time	Dest. in Obstacles
Global (ROS build-in)	11.25	26.34	45.48	0.69	Report failure
Carrot	10.97	Failed	Failed	NAN	Approach point
DStar	26.77	Failed	Failed	NAN	Calculate, and report failure
RAStar	12.06	17.93	48.63	0.05	Report failure
RRT	12.48	19.65	44.34	0.28	Trapped in a loop

This paper designs a system that implements Simultaneous Localization and Map building and navigation of a smartphone remotely controlled vehicle, with the terminal vehicle and 3D LiDAR presented in a simulated ROS environment.

The biggest shortcoming of this system is that no physical production is realized, and all communications and map buildings are tested in virtual environments. Moreover, limited by the author's level, the optimization effect of SLAM+ and RRT algorithm in the paper is limited, and there are loopholes such as scene localization misalignment and planning caught in a dead loop, which is subject to subsequent improvement.

▲ End.