Realization of Outdoor autonomous navigation based on RTAB-MAP

指導教授：邱偉育　　組別：A90　　組員：許晏誠、張瀚、何立平

1. Abstract

Self-driving has been a prevailing topic of research for years. Preceding research has shown promising result within indoor environment while outdoor application has faced some hindrances, such as complexity and noises of the outdoor condition. Thus, in this paper, we'll present a simple framework of autonomous driving in an unknown outdoor space for further research on real-world on-road implementation.

2. Hardware setup

- **Hardware Architecture**

The hardware setup is as Fig. 1 showed. The architecture is based on a 4-wheel RC car which provides us with sufficient power for carrying purposes. The RC car uses Ackerman steering method which differs our model from other differential robots. The model is equipped with RPLidar A2 and a Realsense D435i. Lidar allows the robot to detect the obstacle all around and also perform refinement for visual SLAM. The D435i works as the "eye" of the robot which enables it to see the obstacle at front and provide the image data for RTAB-MAP to build the map and compute the odometry. Our software system communicates with the RC model through an Arduino controller design.

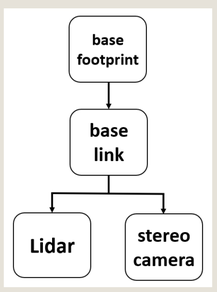

Fig. 1

Fig. 2

- **Controller Design**

The reason why we use Arduino to control the robot is that we can't control the robot by computer directly. If we want to make our robot move, we need an embedded system to send the movement signal to the remote controller. And the embedded system we used is Arduino. The Fig. 3 shows how the controller works.

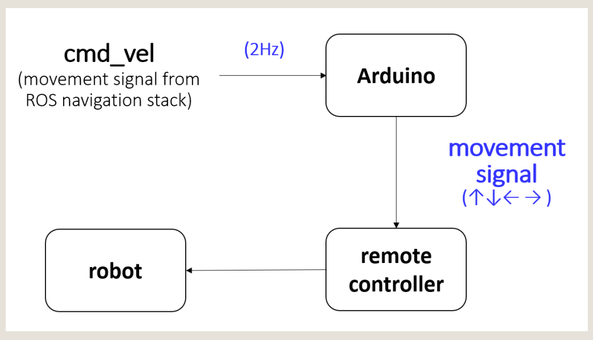

Fig. 3

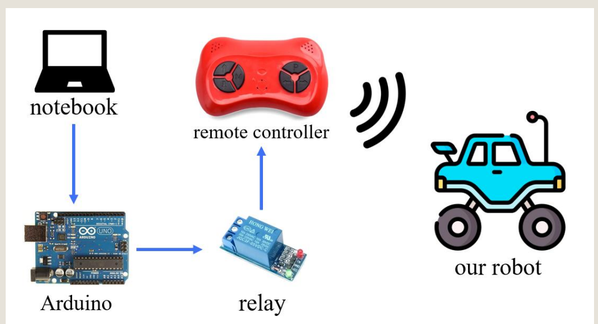

Fig. 4

3. RTAB-MAP

V-SLAM can be illustrated in five blocks in Fig. 10: Initialization, Visual odometry, loop closure, optimization and Mapping.

- Initialization of the global coordinate system enables the system to correctly identify the sensor location within the coordinate system.
- Visual odometry provides the information of the camera's motions and poses.
- Loop closure can judge whether the robot passes through the same place by image matching.
- Optimized pose estimation will be utilized to locate the camera within the system coordinate if loop closures happen.
- Mapping projects the point cloud created from the images' feature points to the grid map.

RTAB-MAP in Fig[9] consists of several independent parts: the sensors, TF, odometry and rtabmap's core which performs the loop closing, graph optimization and map building with the memory utilization of WM and LTM. Stereo odometry was used as our odometry source since it won't be affected by sunlight as severe as RGB mode. Parameters for RTAB-MAP were tuned to work best with our testing conditions. In our system RTAB-MAP plays an important role in producing odometry and static maps for navigation.

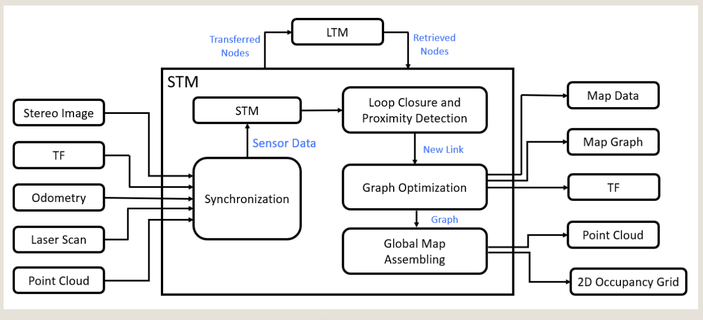

Fig. 9

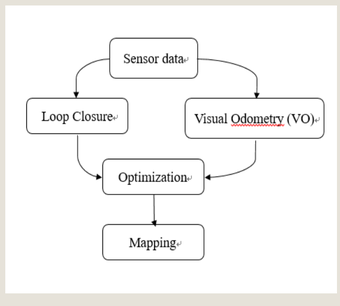

Fig. 10

4. ROS Navigation

ROS navigation stack, in Fig. 8, provides a structure that takes the static map and odometry from RTAB-MAP, sensor data, TF and the goal in the form of coordinate and orientation as input while the output is a combination of linear/angular velocity which will then proceed to our controller. The core of navigation is the "Move_base" that is composed of four components: Global/Local planner, Costmap and recovery behavior. Due to the use of the Ackerman model, the default base_local_planner and Navfn are substituted with teb_local_planner and global_planner that support car-like models and provide orientation targets. The recovery behaviors are designed for car-like robots specifically since our robot cannot perform rotated actions.

- Costmap: The map that the robot will see. It holds the information of the obstacles in the form of grids on the map.
- Global_planner(Global_planner): Global planner will read the coordinate from the '/goal' topic and compute the global path based on global costmap.
- Local_planner(TEB_local_planner): The local planner receives the global path from the global planner. It tends to follow the global path but compute the detailed movements depending on the capacity of different robots.
- Recovery_behavior(stepback_and_steerturn_recovery): When the trajectory from the local planner is not feasible in real-time, the move base will turn to recovery behavior to escape from the current state to compute another trajectory.

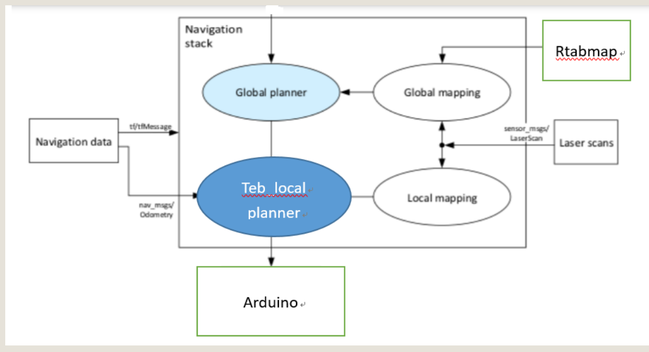

Fig. 8
Navigation stack

5. Result & Analysis

A . Gazebo Simulation

A world includes maze and outdoor environments are created for trials. Large errors occur in simulation due to the lack of visual features. However, short distance tests still showed a precise result.

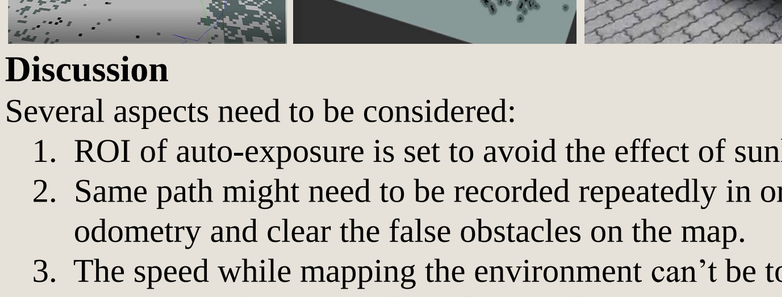

Fig. 5
Gazebo simulation and result

B . Real-world Environment

The tests in the real-world were taken in the periphery(open space) and the square(semi-open space) of the Delta building. In comparison with simulation, real-world application showed a precise odometry result with only 30cm of errors. In semi-open space, the odometry will slightly jump due to similar visual features. The navigation trials showed a promising result for the goal set at the front. However, more computations are needed for backward goal and sometimes will be stuck in the local minimum.

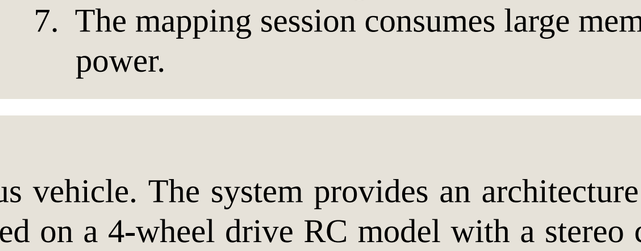

Fig. 6
mapping result of peripheral and square of Delta

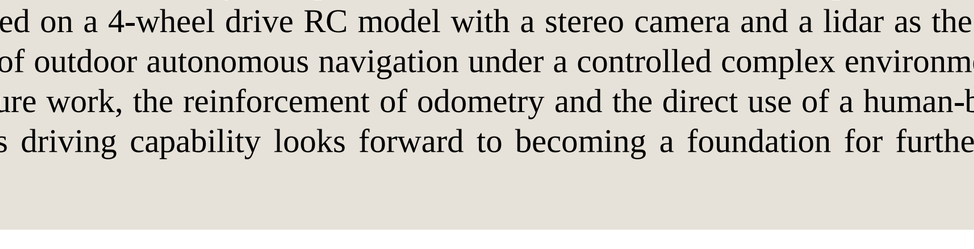

Fig. 7
Navigation results with obstacle avoidance. Blue circles stand for the obstacle. The red path is the local plan. The last picture demonstrates the obstacle avoidance with the car being blocked.

Discussion

Several aspects need to be considered:

1. ROI of auto-exposure is set to avoid the effect of sunlight.
2. Same path might need to be recorded repeatedly in order to either fix the drift in odometry and clear the false obstacles on the map.
3. The speed while mapping the environment can't be too fast in case that the computing doesn't catch up the speed.
4. The height of the sidewalk is sometimes too low to be detected by camera and lidar.
5. Since the controller frequency is retained to 2Hz, the robot will not avoid fast moving obstacles like bicyclers in time.
6. Lanes and road signs aren't taken into consideration. For on road implementation, integration of computer vision is needed.
7. The mapping session consumes large memory, time, computation resources and power.

6. Conclusion

In this project, we rolled out an implementation of system integration for an autonomous vehicle. The system provides an architecture which is composed of visual SLAM, RTAB-MAP, and ROS navigation stack with modification plugins. The proposed model is based on a 4-wheel drive RC model with a stereo camera and a lidar as the sensors. The setup along with some modifications with respect to the result of trials can realize the purpose of outdoor autonomous navigation under a controlled complex environment. After the tests, defects, like detection failure of lower obstacles or odometry drifting, were found. In future work, the reinforcement of odometry and the direct use of a human-based map on navigation can be carried out. As a result, our proposed robot with a promising autonomous driving capability looks forward to becoming a foundation for further research or application with self-driving cars on campus.