Introduction

Shizeng

I. System Overview

This demo is based on the Linux and ROS operating system and I finished the functions using the C++ and Python languages. The goal of the demo is to simulate a car to find the treasure under the unknown conditions. The function nodes structure is below:

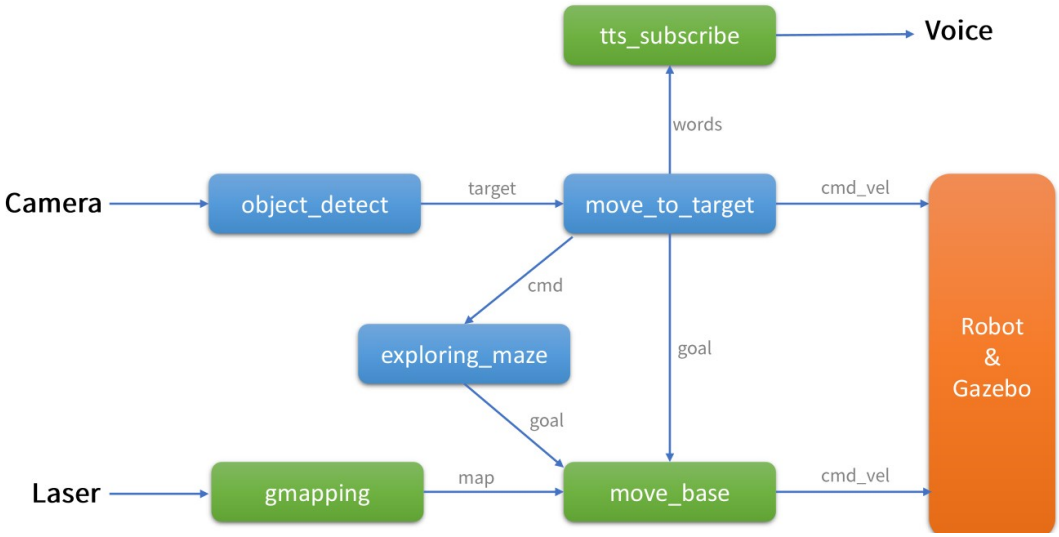


Fig 1: function nodes structure

Two sensors to calculate the map information and send these to object_detect node and gmapping node separately. The function of object_detect node is to identify the object whether it is the treasure which we want to find. Gmapping is a open source function package and I use it to do the mapping. After that gmapping node will send the map information to move_base node. For the move_to_target and exploring_maze nodes, these two are used to send the goal information to the move_base node which means to tell the move_base how far away the car is to the goal. Finally, the move_base can control the position and speed of the car to get the treasure.

II. System Functions

For the object_detect node, the function of this node is to detect the treasure. The basic idea is to use the colour difference between the object and the background. I use OpenCV to get this function. After the detect, I can get the useful information, one thing is the position of the goal and another one is the size.

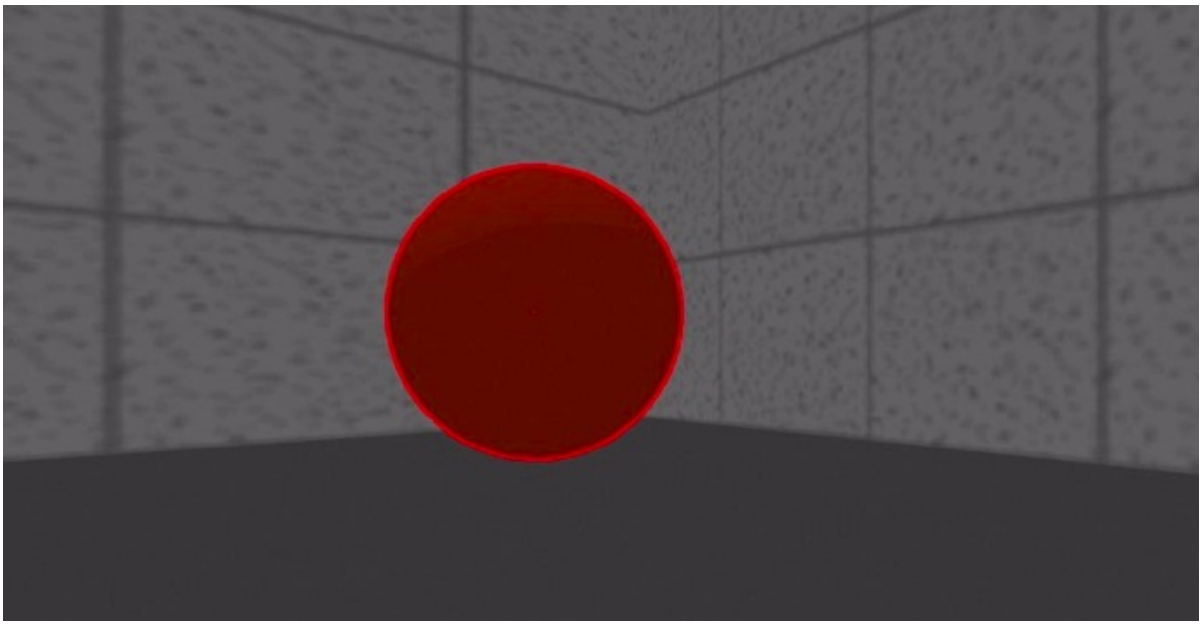
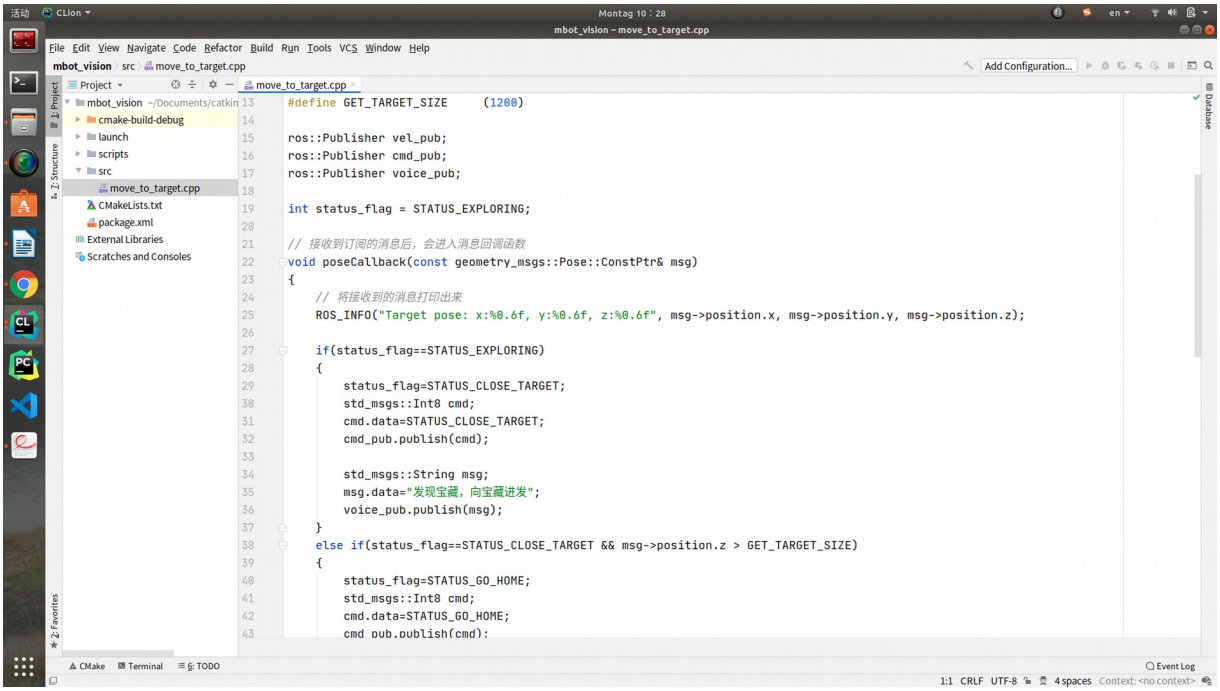


Fig 2: result of the detect

For the move_to_target node, the function is to control the car to close the goal. Its basic idea is to use the compare the size of the find one and the real object. Here, I set a range of the true size like below, then I can use this function to control the car keep a reasonable distance before the object.



```
#define GET_TARGET_SIZE     (1200)

ros::Publisher vel_pub;
ros::Publisher cmd_pub;
ros::Publisher voice_pub;

int status_flag = STATUS_EXPLORING;

// 接收到订阅的消息后，会进入消息回调函数
void poseCallback(const geometry_msgs::Pose::ConstPtr& msg)
{
    // 将接收到的消息打印出来
    ROS_INFO("Target pose: x:%0.6f, y:%0.6f, z:%0.6f", msg->position.x, msg->position.y, msg->position.z);

    if(status_flag==STATUS_EXPLORING)
    {
        status_flag=STATUS_CLOSE_TARGET;
        std_msgs::Int8 cmd;
        cmd.data=STATUS_CLOSE_TARGET;
        cmd_pub.publish(cmd);

        std_msgs::String msg;
        msg.data="发现宝藏，向宝藏进发";
        voice_pub.publish(msg);
    }
    else if(status_flag==STATUS_CLOSE_TARGET && msg->position.z > GET_TARGET_SIZE)
    {
        status_flag=STATUS_GO_HOME;
        std_msgs::Int8 cmd;
        cmd.data=STATUS_GO_HOME;
        cmd_pub.publish(cmd);
```

Fig 3: move_to_target node

For the exploring_maze node, it identifies three status: STATUS_EXPLORING,STATUS_CLOSE_TO_TARGET and STATUS_GO_HOME. This node is like a control server. The code is below:

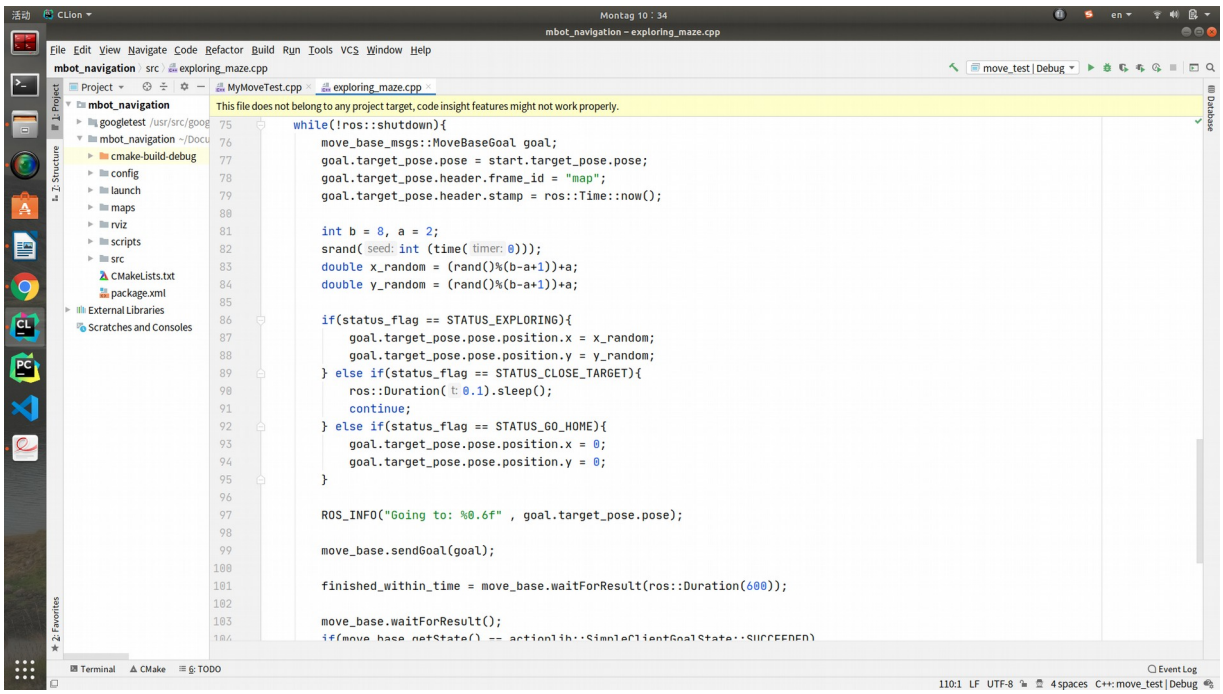


Fig 4: exploring_maze node

Based on these nodes, the treasure can be found.