实验报告

姓名：廖嘉辉
时间：7.24-8.6

【实验目的】
ROS 学习

【实验过程】
1. 获取激光雷达数据
   查找激光雷达消息包格式

sensor_msgs/LaserScan Message

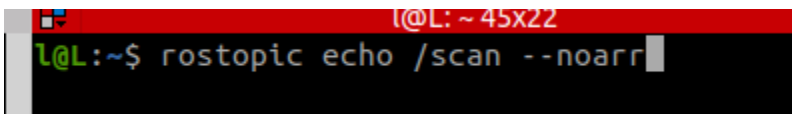
File: sensor_msgs/LaserScan.msg

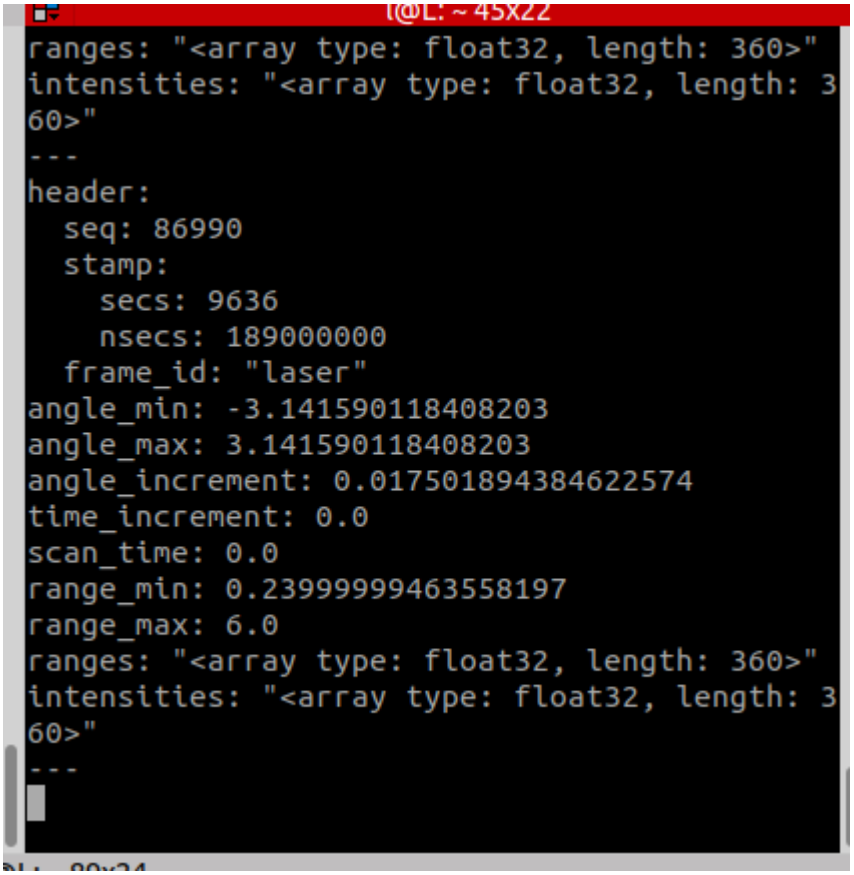
Raw Message Definition

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

Compact Message Definition

用“rostopic echo /scan --noarr”查看/scan 话题内的消息，
“--noarr”由于折叠数组避免刷屏





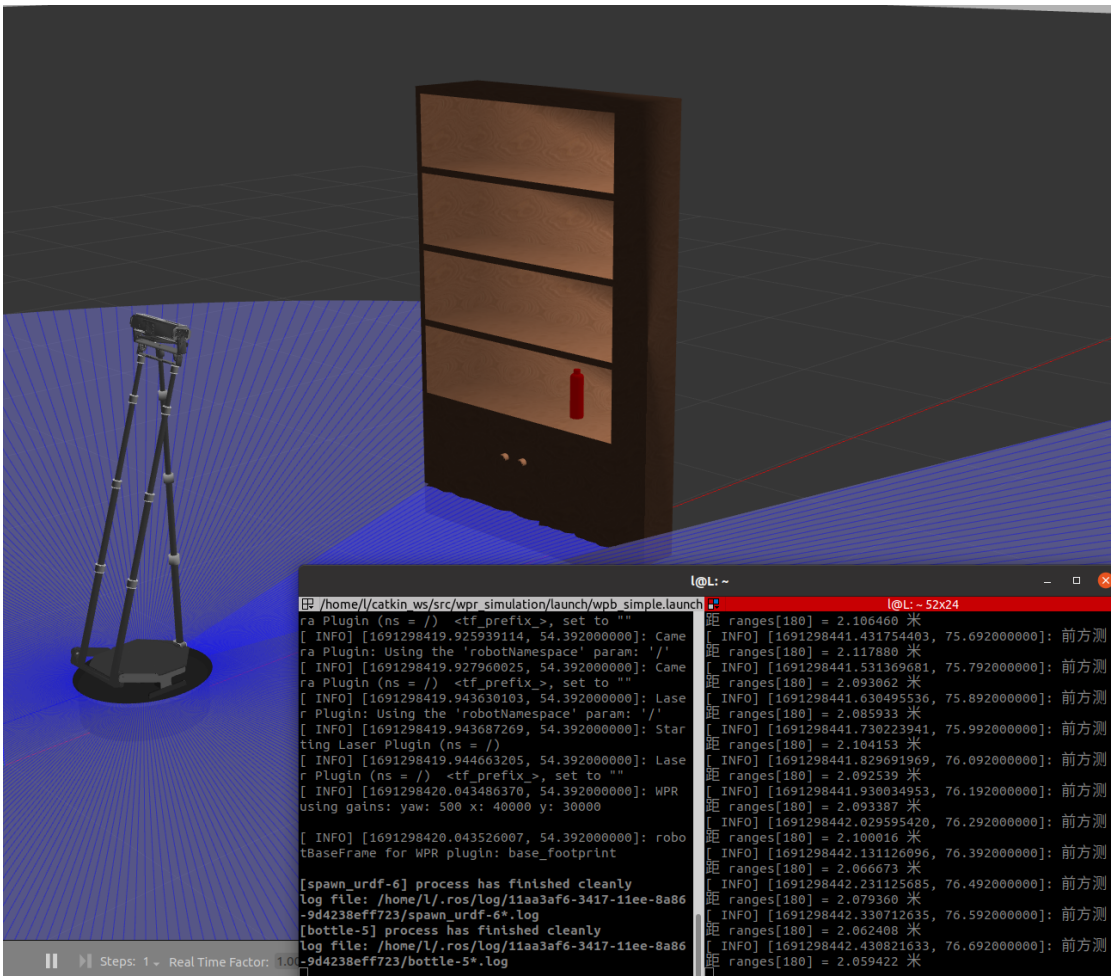
2. 实现障碍物躲避

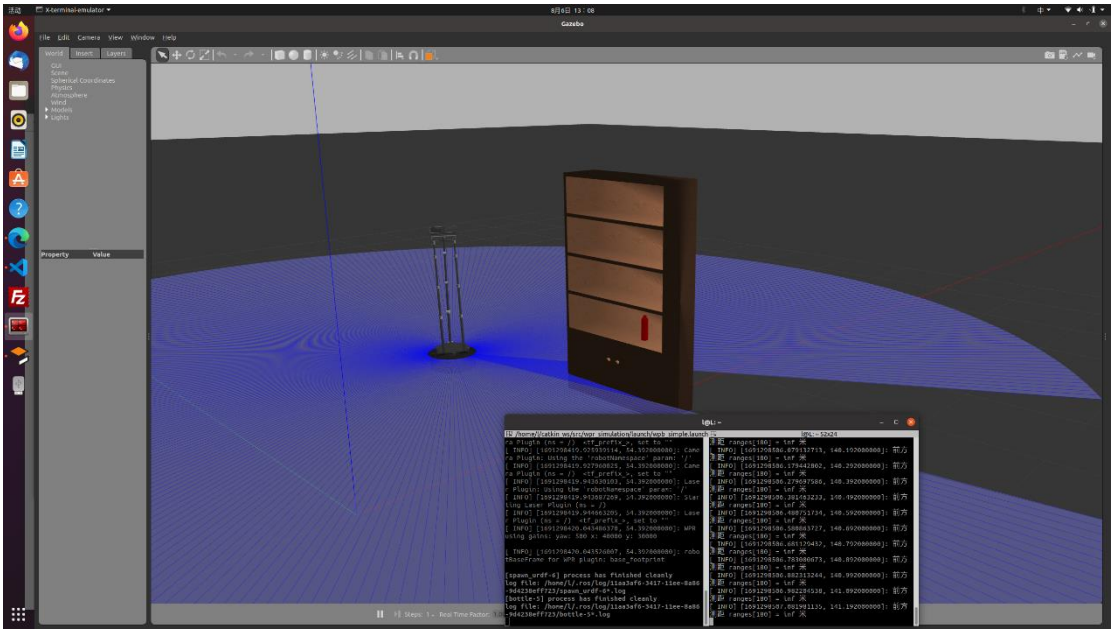
lidar_node.cpp

src > lidar_pkg > src > lidar_node.cpp > LidarCallback(const sensor_msgs::LaserScan)

```cpp
#include <ros/ros.h>
#include <sensor_msgs/LaserScan.h>
#include<geometry_msgs/Twist.h>//速度消息格式头文件

ros::Publisher vel_pub;//声明全局变量
int nCount =0;

void LidarCallback(const sensor_msgs::LaserScan msg)
{
    float fMidDist = msg.ranges[180];
    ROS_INFO("前方测距 ranges[180] = %f 米",fMidDist);

    if(nCount>0)
    {
        nCount--;
        return;
    }//延长旋转时间，确保不会撞到障碍物


    geometry_msgs::Twist vel_cmd;//构建速度控制消息包
    if(fMidDist<1.5)
    {
        vel_cmd.angular.z=0.3;
        nCount=50;
    }
    else
    {
        vel_cmd.linear.x=0.05;
    }
    vel_pub.publish(vel_cmd);

}
```

```cpp
int main(int argc, char  *argv[])
{
    setlocale(LC_ALL,"");
    ros::init(argc,argv,"lidar_node");

    ros::NodeHandle n;
    ros::Subscriber libar_sub=n.subscribe("/scan",10,&LidarCallback);
    vel_pub=n.advertise<geometry_msgs::Twist>("/cmd_vel",10);//发布速度控制话题

    ros::spin();



    return 0;
}
```





3. 获取 IMU 消息包格式

File: sensor_msgs/Imu.msg

Raw Message Definition

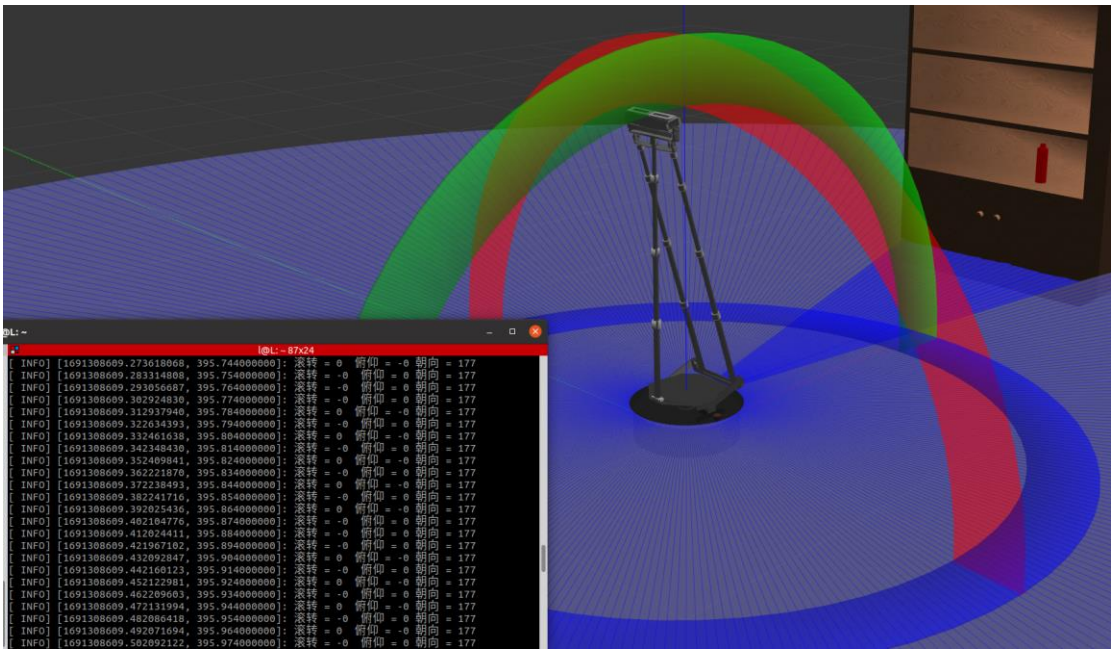
```
# This is a message to hold data from an IMU (Inertial Measurement Unit)
#
# Accelerations should be in m/s^2 (not in g's), and rotational velocity should be in rad/sec
#
# If the covariance of the measurement is known, it should be filled in (if all you know is the
# variance of each measurement, e.g. from the datasheet, just put those along the diagonal)
# A covariance matrix of all zeros will be interpreted as "covariance unknown", and to use the
# data a covariance will have to be assumed or gotten from some other source
#
# If you have no estimate for one of the data elements (e.g. your IMU doesn't produce an orientation
# estimate), please set element 0 of the associated covariance matrix to -1
# If you are interpreting this message, please check for a value of -1 in the first element of each
# covariance matrix, and disregard the associated estimate.

Header header

geometry_msgs/Quaternion orientation
float64[9] orientation_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance # Row major x, y z
```

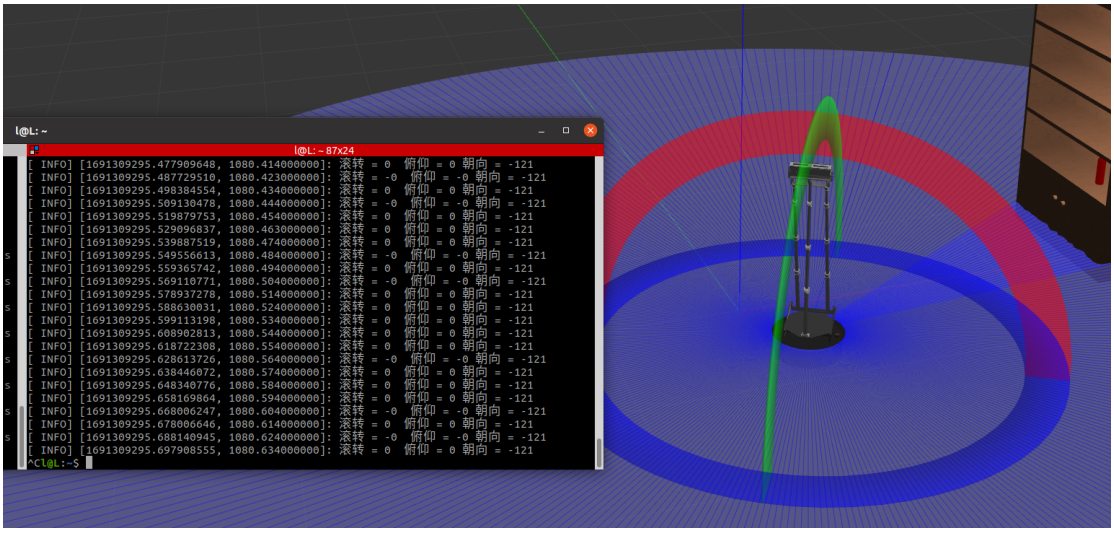
4. 获取机器人空间姿态

src > imu_pkg > src > imu_node.cpp > IMUCallback(sensor_msgs::Imu)

```cpp
#include<ros/ros.h>
#include<sensor_msgs/Imu.h>//包含IMU消息头文件
#include<tf/tf.h>


void IMUCallback(sensor_msgs::Imu msg)//IMU回调函数
{
    if(msg.orientation_covariance[0] < 0)
        return;//判断四元数的协方差矩阵的第一个数值进行判断，确定四元数是否存在
    tf::Quaternion quaternion(
        msg.orientation.x,
        msg.orientation.y,
        msg.orientation.z,
        msg.orientation.w
        );//将消息包的四元数转换为四元数对象



    double roll,pitch,yaw;
    tf::Matrix3x3(quaternion).getRPY(roll,pitch,yaw);//四元数转为欧拉角


```

```cpp
    double roll,pitch,yaw;
    tf::Matrix3x3(quaternion).getRPY(roll,pitch,yaw);//四元数转为欧拉角


    roll=roll*180/M_PI;
    pitch=pitch*180/M_PI;
    yaw=yaw*180/M_PI;//弧度换为角度

    ROS_INFO("滚转 = %.0f  俯仰 = %.0f 朝向 = %.0f",roll,pitch,yaw);
}


int main(int argc, char  *argv[])
{
    setlocale(LC_ALL,"");//避免显示中文变乱码
    ros::init(argc,argv,"imu_node");//初始化节点

    ros::NodeHandle n;
    ros::Subscriber imu_sub=n.subscribe("/imu/data",10,IMUCallback);
```





【实验结果】

1. 了解机器人避免碰撞控制方法
2. 了解机器人空间姿态获取方法