LoomoRescue: Application of Multi-sensor Fusion in Simultaneous Localization and Mapping and Path Planning

Submitted by

Zhouyao YU

Thesis Advisor

Prof. Dr.-Ing. Denys J.C. MATTHIES
Sven Ole SCHMIDT, M.Sc.

INFORMATION TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
TECHNICAL UNIVERSITY OF APPLIED SCIENCES LÜBECK

A thesis submitted in fulfillment of the requirement to the degree
Bachelor of Science (B.Sc.)

2022

TECHNISCHE HOCHSCHULE LÜBECK

Bachelor thesis

LoomoRescue: Application of Multi-sensor Fusion in Simultaneous Localization and Mapping and Path Planning

Submitted by: Zhouyao Yu

Department: Electrical Engineering and Computer Science

Degree program: Information Technology

First examiner: Prof. Dr. Matthies

Date handing out: 15th March 2022

Date handing in: 15th June 2022

A. Hanemann

(Prof. Dr. Andreas Hanemann)
Head of Examination Board

Task description:

Robots that assist humans are an age-old vision and still a trend in the research field of human-robot interaction. In this thesis the Loomo should be used as a rescue bot. Previously, Loomo was used to patrol the entire floor, while the built-in attitude sensor for motion control was utilized in combination with the electromagnetic positioning system for navigation. However, this has certain limitations such as when deployed in a real-world setting. Loomo cannot detect moving obstacles with the previously deployed navigation technique. Although Loomo can build the map and plan the path using the external positioning system, this capability is compromised by the low range and coverage. Therefore, computer vision techniques could be utilized to complement navigation. Moreover, Loomo's ultrasonic sensor for close range detection could also be incorporated to gain accuracy. With four sensors working at the same time, Loomo should be able to provide superior localization in order to navigate through rooms such as dangerous indoor environments and to automatically create a map. The key questions are; how to sensibly fuse all sensors so that the lowest possible error rate is generated, how to derive a complete map, such as a floor plan and how to navigate Loomo through space with moving obstacles. The thesis begins with a literature search on assistive robots and rescue robots. In addition, sensor fusion approaches, visual computing algorithms to detect objects and environments, and spatial navigation algorithms should be reviewed. The main part of this work is the implementation, which is completed with a short evaluation that demonstrates the performance of the implementation.

Prof. Dr. Matthies

TECHNISCHE HOCHSCHULE LÜBECK

Department of
Electrical Engineering and Computer Science
Dean's Office - Board of Examiners

Statement on the bachelor thesis

I assure you that I wrote the work independently, without outside help.

Only the indicated sources has been used in the preparation of the work.
The text or the sense of the text are marked as such.

I agree that my work may be published, in particular that the work may be submitted to third parties for inspection or that copies of the work may be made for forwarding to third parties.

15.06.2022
Date

Zhou yao 周尧
Signature

Declaration of Authorship

I, Zhouyao Yu, declare that this thesis titled, "LoomoRescue: Application of Multi-sensor Fusion in Simultaneous Localization and Mapping and Path Planning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly conducted while in candidature for a degree at Technical University of Applied Sciences Lübeck.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Zhouyao Yu 余宙曉

Date: 15.06.2022

TECHNICAL UNIVERSITY OF APPLIED SCIENCES LÜBECK

Abstract

Department of Electrical Engineering and Computer Science

Bachelor of Science (B.Sc.)

LoomoRescue: Application of Multi-sensor Fusion in Simultaneous Localization and Mapping and Path Planning

by Zhouyao YU

Entering a room on fire to search for the injured is a very risky and time-consuming task. In addition, the fire environment is dangerous and volatile, with many areas impassable due to changing conditions. Although numerous designs of remote controlled fire rescue robot are proposed, most of them totally depend on the operator to detect the environment. If multiple sensors are deployed on robot and contribute to perceive the environment, operators' workload will be reduced and thus improve the efficiency. In this paper, a fire rescue robot is designed to discuss this idea.

Acknowledgements

Put your acknowledgements here! This is an example:

Acknowledgments by Zhouyao Yu I want to give thanks to my mentor Sven Ole Schmidt. During the graduation design, he shared his knowledge and experience with me and cheer me up when I was upset. I am also thankful to Denys Matthies. He taught me detailed skill about scientific writing and give me technical support on my project.

I especially feel grateful to my teacher Zhengpo Li in ECUST. During three years of study in ECUST, he enrolled me in competitions and projects of hardware design, during which I almost learned all knowledge and technique used in the project of my bachelor thesis from him.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

UWB	**U**ltra **W**ide **B**andwidth
UDP	**U**ser **D**atagram **P**rotocol
RGB	**R**ed **G**reen **B**lue
RGB-D	**R**ed **G**reen **B**lue - **D**epth
PID	**P**roportional **I**ntegral **D**erivative
OCU	**O**perating **C**ontrol **U**nit
IMU	**I**nertial **M**easurement **U**nit
SLAM	**S**imultaneous **L**ocalization and **M**apping
OP	**O**perator

1 INTRODUCTION

1.1 Motivation

Fire rescue robots have been studied and matured for many years, and among them there are many products that are already in use and have achieved results[1][2]. In fires, these robots, remotely controlled by firefighters, perform tasks such as personnel search, environmental assessment, and evacuation guidance. They accomplish their jobs well, reducing rescue risks and increasing efficiency. However, for each fire, firefighters are confronted with a completely new terrain environment. In other words, while searching and rescuing, the robot operator also has to take on the prerequisite task of exploring the indoor environment, adding to the rescue burden.Further, the complex and changing environment in a fire scene makes it extremely difficult to complete a rescue with only the operator's memory of the terrain, not to mention getting lost due to unfamiliarity with the terrain.Additionally, factors such as high temperatures in a fire scene can affect the quality of communication. Remote control signals are often lost or delayed, which can cause the robot to lose control. Therefore, if collision prevention, path planning, and environmental mapping can be added to search and rescue robots, then not only can accidents be prevented when misuse or poor communication quality occurs, but ergonomics will also be greatly

enhanced, thus reducing the workload of robot operators and improving rescue efficiency.

Further, with the development of advanced control technology, autonomous algorithms for robots such are becoming more and more mature. Combining them with rescue robots would be a promising direction. By allowing robots to search and rescue on their own to reduce the human burden, even if the efficiency is not as good, it is possible to expand the search and rescue range and improve the success rate by deploying multiple simultaneous search and rescue. Even automated personnel search and rescue and evacuation guidance procedures can be deployed on home robots, allowing them to evacuate affected people from the danger zone as early as possible before the fire department arrives and before the fire spreads, further reducing damage.

1.1.1 Goal

In this project, an Android-based software is expected to be developed to solve the above mentioned problem. It has a host program and a robot control program, and both can communicate in at least one way. On the host side, the user can get the robot status and sensor information, and can also give movement commands to the robot. At the robot side there is a motion control system, a sensor bus and a communication module. With the two terminals together, the robot is able to explore unknown environments. In addition, some path finding algorithms and robot path finding functions can be tried to verify the feasibility of automated search and rescue and guided evacuation.

1.2 Organization

This thesis will discuss this problem in six parts. Chapter 2 is an introduction to the relevant work. Chapter 3 specifies the project requirements. The process made by former developer will also be introduced. Chapter 4 explains the relevant theory, including those of the essential technology applied in this project. Chapter 5 describes the selected and alternative design of solution. Chapter 6

describes the systematic implementation of each module. Chapter 7 is the evaluation of system function. Chapter 9 introduce some extra work on this project. Chapter **??** draws a conclusion on this project and share some ideas about future work.

2 RELATED WORK

2.1 Portable fire evacuation guide robot system

Over the past few decades, many rescue robots have been invented. One of them is a remote-controlled fire rescue robot proposed by in 2009[3], which is inspiring. The application scenario flow is shown in figure 2.1[3].

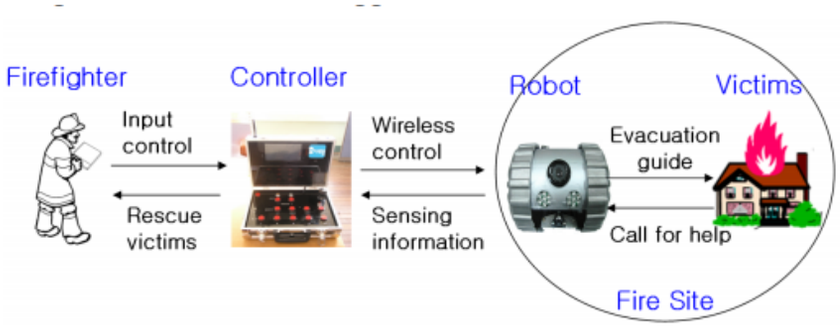

FIGURE 2.1: Flow diagram of system

Firefighters deploy the robot and control its actions by remote control, and then guide the evacuation in the fire scene. At the same time, firefighters receive feedback from the robot in the form of image information and sensor data. In addition, the firefighter is able to communicate with the victims and guide them to evacuate through voice interface and indicator lights. In the actual functional

implementation, the authors used Bluetooth, Video RF and Voice RF to communicate between the controller and the robot. This multi-media approach allows for improved communication robustness and interference immunity. At the controller side, two modes of operation were designed, one using the control box and the other using the touch screen control. The control box is a box that contains a display, a joystick and buttons, where the joystick can directly control the robot, while the buttons contain some integrated commands. This makes it easier for firefighters to operate with gloves. In the other model, a GUI contains all the functions and is more portable. In the mechanical design, the robot uses a fire-resistant aluminum housing and a dual-wheel plus auxiliary wheel design to make its shape small enough and flexible. On the software side, a extendable operation system is deployed to coordinate the work of the sensors on the robot. When more specialized sensors are needed, this system makes it easy to make them work.

In the authors' evaluation, the robot passed mechanical performance tests such as heat and water resistance well, and the controller was sufficient to assume communication between the firefighter, the robot and the victim. As seen in the paper, Bluetooth is able to assume the communication medium in the fire scene to some extent. Moreover, the redundant design of both Bluetooth and RF communication media can further enhance the robustness of the system, and such a design is worth learning from. However, the system has not been tested in a real fire. Considering the hot flames in the fire field, the terrain obstruction and other factors can affect the quality of the communication system, there may actually be a delay or loss of signal, which then leads to the robot losing control. Therefore some algorithms to cope with this situation could be considered, or algorithms for automatic path finding could be tried.

2.2 A Search-and-Rescue Robot System for Remotely Sensing the Underground Coal Mine Environment

In a design for a coal mine rescue robot proposed[4], some features are designed to prevent unexpected situations such as collisions. In this project, a remotely controlled robot is used to explore a coal mine shaft and collect environmental information. The electrical system consists of four components, sensor system, battery, motion control unit, operating control unit (OCU) and communication system. The OCU has an electronic compass, gyro and two wheel encoders and four infrared sensors. The infrared sensors are used to measure the distance of the robot from the wall to prevent impact. The rest of the transmission units are used to deduce the motion trajectory and form part of the positioning function. Further, the authors use the image information from the camera to correct the accumulated errors of the sensors to improve the localization performance. The communication architecture consists of an Operating Control Unit (OCU), a Relay box and a Robot Electric box, which are connected to the Relay box and the Robot by optical fiber and its adapter, and to the OCU by WIFI. TCP/IP protocol is used in communication, and the overall structure of communication is based on the network, so every unit communicate with the others.

What is noteworthy in this paper is that in addition to the sensor modules for the measurement tasks, the robot also has a dedicated motion control system. Although the robot is mainly controlled by a human and has a solid and reliable communication with fiber optics and WIFI, it is very difficult to understand a new environment only through the images of the on-board cameras, which leads to misjudgment of the environment, not to mention that humans can also make operational errors. The anti-collision function and odometer function of this motion control system give the robot a certain degree of local autonomy, allowing it to have better performance in motion.

3

REQUIREMENT ANALYSIS

The project is developed based on Loomo robot. It is a combination of Segway Robotics' self-balancing vehicle and a companion robot. In addition to being a vehicle and a personal robot, the Loomo can also be used for personal programming experiments when set to developer mode. The whole system consists of two applications, one is the robot control application and the other is the host computer application. The host computer application has a GUI that allows the operator to control the robot and to see the image information and data coming back from the robot. The robot control application motion control system, sensor bus and communication module. The system operates in the following scenarios.

3.0.1 Scenario

As shown in figure 3.1, firefighters arrive when the fire breaks out, deploy Loomo at the door and turn on the host computer. The OP connects to Loomo via Bluetooth or WIFI. After successful connection, the GUI displays the screen coming from the Loomo camera and shows the status information of the Loomo. The operator then remotely controls the movement of Loomo directly to make it explore the interior of the room, and the objects in front of the robot can be seen

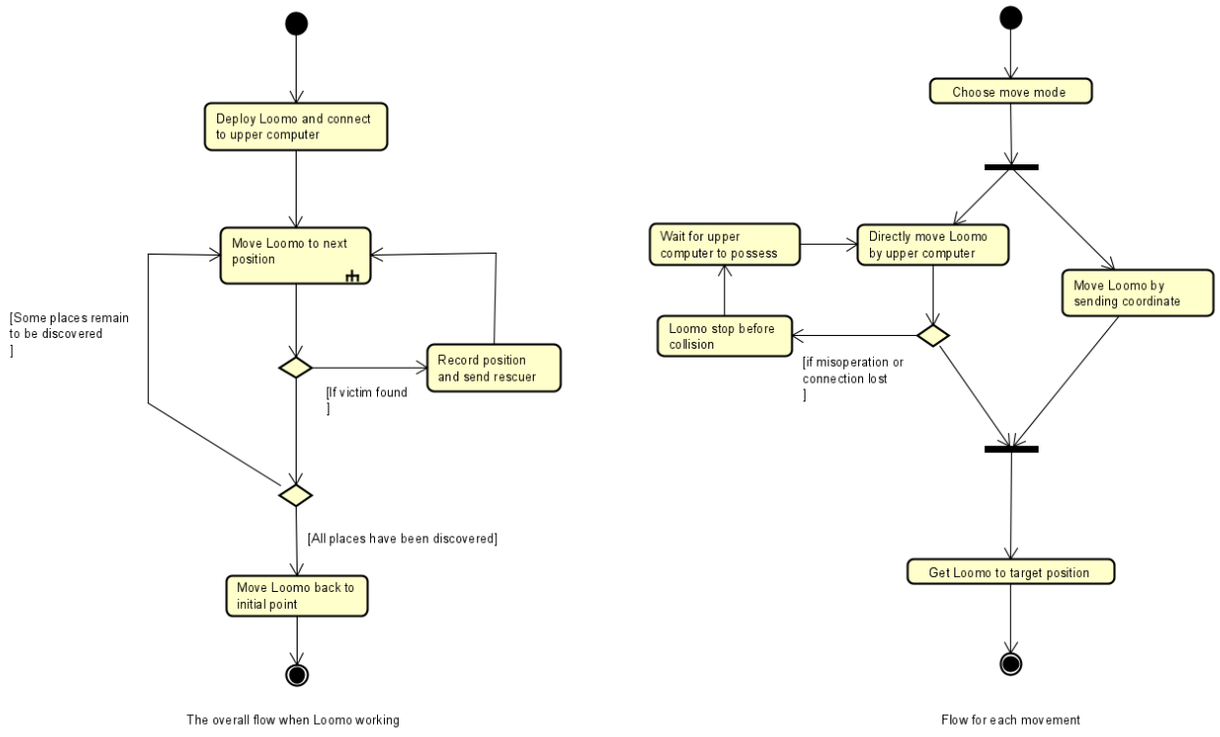

FIGURE 3.1: Activity diagram of rescue

clearly through the transmitted screen. Based on the coordinate information returned by the positioning system in the room and Loomo's trajectory projection function, its location can be displayed on the map. As Loomo went deeper, the signal received interference, and the firefighter's operation began to experience a delay. Facing the obstacle in front of it, it failed to stop in time, and Loomo's collision prevention function took effect and stopped in front of the obstacle. Since the delay was too large to directly control Loomo remotely the operator chose to use another command, he sent a coordinate to Loomo according to the map, and then Loomo moved to that coordinate by itself with the help of the positioning system, sending back image information intermittently during the period. Along the way, if a casualty is found, the operator can mark the location on the map based on the current coordinates. When the room is traversed, the operator can control Loomo or let it return to the starting point by itself.

3.0.2 Functional Requirements

According to the scenario, the following functional requirements are derived. These requirements define the functionality of a software system or component, and are the functions and services that a system needs to provide. Functional requirements contain functional descriptions, technical details, and other elements that describe the functionality that the system wishes to achieve.

1. A host computer program with GUI capable of acting as a remote control is to be implemented, which is able to
   (a) Connect to Loomo via WIFI and Bluetooth.
   (b) Directly move Loomo by using button on GUI
   (c) Indirect control Loomo's movement by sending coordinates through the GUI.
   (d) Show Loomo's position on GUI
2. A program deployed on Loomo which enable robot to
   (a) Communicate with host computer
   (b) Subscribe data from external position system via network
   (c) Move Loomo accurately to a target position
   (d) Collision avoidance

3.0.3 Nonfunctional Requirements

Non-functional requirements define requirements beyond meeting business needs and are constraints in program development. These constraints are used to ensure additional performance of the program such as extendibility, portability, flexibility, etc. The non-functional requirements for this project are as follows.

1. The whole project should based on android platform. The android sdk version of Loomo must be lower than 5.1.

2. Use Loomo as rescue robot.

4

BASIC PRINCIPLE

4.1 Ultrasonic sensor

One of the most important component is Ultrasonic sensor. An ultrasonic sensor is a device that converts ultrasonic signals into electrical signals. Ultrasound is a sound wave with a frequency of more than 20 kHz, which travels at 343 meters per second in air at 20 degrees Celsius. It is usually designed to measure the distance from an object to an obstacle along with a transmitter that emits ultrasonic waves at 40 kHz[5]. The principle is to measure the time interval between the emitted and received ultrasonic signals. When working, it will first emit a train of ultrasonic pulses, which will be reflected back when it touches the object. The reflected back ultrasonic signal will be received by the sensor and converted into an electrical signal. By recording the time when the sound wave is emitted from the transmitting end and the time when the sound wave reflected back from the object is received, the time interval of the sound wave flying in the air can be calculatedwhich is also called time of flight(ToF). Following that, the distance of the object can be calculated from the speed of sound[6].

4.2 inertial measurement unit, IMU

The inertial measurement unit is a sensor for obtaining attitude information. It basically consists of three linear accelerometers which measure acceleration and three rate gyroscopes which measure angular rotation rate.[7] The accelerometer is essentially a force sensor, which measures the force in each direction by squeezing a piezoelectric crystal and converting it into acceleration. The angular velocity sensor obtains angular velocity information through an electronic gyroscope. An IMU containing both of these sensors is known as a 6-axis IMU. although it is sufficient to describe attitude, the angular velocity data drifts as errors accumulate. To address this situation, the 9-axis IMU was designed to include a three-axis geomagnetic sensor in addition to the 6-axis, which corrects the angular velocity sensor by the absolute pointing of the geomagnetic field to give accurate readings.[8]

4.3 Ultra wide band based multi anchor localization system

makes use of the electromagnetic wave ToF to measure the distance. the TOF ranging method is a two-way ranging technique, which mainly uses the time of flight of the signal between two transceivers to measure the distance between nodes. When working, the Double-Sided Two-Way-Ranging technique is used. This is done to eliminate the error in calculating the ToF caused by the Clock reference difference between the transceivers. The signal transceivers in the system are all the same, but they are divided into two roles: one is a point with a known position, called the anchor point, and the other is a point with an unknown position, called the tag. When the system works, the coordinates of the tag are located by measuring the distance between the tag and each anchor point.[9]

4.4 PID Control

In order to improve the response performance of the robot, PID control is introduced. PID control is when the output of the system is obtained, and the output is superimposed on the input through proportional, integral and differential operations, thereby controlling the control method of the system. The flow diagram of PID control is shown in figure 4.1 [10].

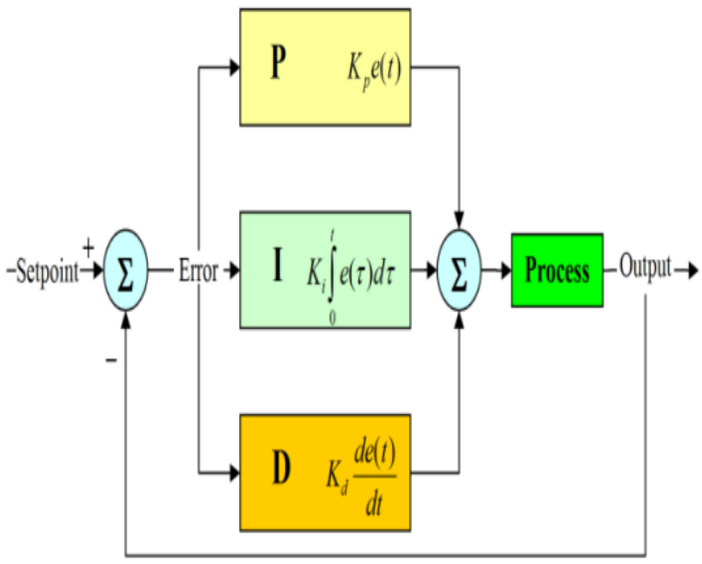

FIGURE 4.1: PID block diagram

A proportional link multiplies the output by a factor, which is then added to the input, and this factor is called the proportional parameter. Proportional control allows fast, timely and proportional adjustment of deviations. Therefore it can improve the control sensitivity, but there is static difference and low control accuracy. If the proportional parameter is larger the proportional effect is stronger, then the dynamic response is faster and the ability to eliminate errors is stronger. However, the actual system is inertial, and the output value changes after the control output changes, and it needs to wait for some time before it changes slowly. The scale of P-parameter should be decided based on the field commissioning according to the system response, and usually the P-parameter is adjusted from large to small to achieve the fastest response without large overshoot as the best parameter.

The integral element is used to eliminate the static difference. The principle is that as long as the error exists, the error is integrated, so that the output continues to increase or decrease until the error is zero, that is, the system is stable. At that moment, the integration stops and the output no longer changes. Thus, the effect of non-differential regulation is achieved. The integration effect is controlled by the integration time and gain, which are the parameters of the integration element. Again, due to system inertia, the integrator effect requires a waiting period. Therefore the integrator effect needs to match the actual system response speed. If the parameters are too small, the system will be time-consuming to stabilize, and conversely, if they are too large, the system will oscillate near the stabilization point. Therefore, it is better to adjust the integration parameters in such a way that stability can be restored as soon as possible without causing oscillations.

The differentiation element differentiates the input and output errors and multiplies them by a factor to add to the input. Its role is to override the control and meet the dynamic indicators of the response. The dynamic indicator refers to the rate of return to steady state after a disturbance caused by a load change or a given adjustment, etc. Unlike the two aforementioned control links that start working after an error has been generated, the differential link starts working when there is a tendency to generate an error. The differential link speeds up the recovery by controlling in this corresponding trend way. Although the differential link makes the response of the system faster, reduces overshoot, and mitigates oscillations, its effect is too pronounced and prone to oscillations to play a dominant role in the control link. Fine tuning should be done after the parameters of the proportional and integral links are determined.

The primary task of any closed-loop control system is to respond to commands stably, quickly and accurately. PID control is to achieve this by adjusting the specific values of the three parameters.

4.5 Simultaneous localization and mapping, SLAM

Simultaneous Localization and Mapping(SLAM) is a technique which focus on navigating without prior knowledge about environment[11]. It create a map through recognition of landmarks in environment where robot has explored. The map consist of robot's position and relative positions of landmarks respect to robot. The data will later used to estimate robot's exact location. The advantage of SLAM is that it can generate geometrically consistent map and position robot and landmark simultaneously. As a result, it suits robot in autonomous mobilization.

The Mathematical description of SLAM is showed in figure 4.2 [12]. This is a Bayes formula that describe one SLAM iteration in probabilistic form.

$$P(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$$

FIGURE 4.2: Probabilistic formula of SLAM

Considered single and accumulative error of sensors, it is impossible to measure the precise coordination through direct observation. The formula shows the possibility of map and position after iterating k time.

- X: a vector description of robot state, which include translation and rotation.
- M: he map consist of vectors recording the positions of all landmark.
- Z: set of all historical landmark observation.
- U: set of all historical motion input of robot like acceleration or rotation. It is used to estimate the position of robot through integration.
- X0: the initial location of robot.

The formula shows the way SLAM works. It demands multiple iterations to generate the map and solve for the current position, and each iteration depends on

results of the former one. Therefore, the result is derived from all historical observation and estimation. For each iteration, it is required to discover coordinate of landmarks, calculate location of robot and record state of robot, including posture, speed and acceleration, for next iteration. With multiple iteration a map was generated through initial point and set of landmarks represented by relative coordinates.

5

MODULE DESIGN

There are Ultrasonic sensor, hall sensor and Inertia measure unit embedded in Loomo . To combine them cooperatively, it is necessarily to make them well organized through a structure. Besides, only with the help of external UWB position system can the robot work properly, so it is also a point to make internal and external sensors unite. This section describes the overall system software framework design and communication structure design. Moreover, the functions of each subpart are explained.

5.1 Overview

In addition to the internal sensors of Loomo, an external positioning system is introduced to improve the positioning accuracy and correct the accumulated errors of the robot attitude sensors, which is the UWB positioning system mentioned above. The positioning system is connected via a network. Therefore, the whole system consists of four parts: Loomo, the host computer, the positioning system and the router. The subparts are connected to each other via Bluetooth or network protocol.

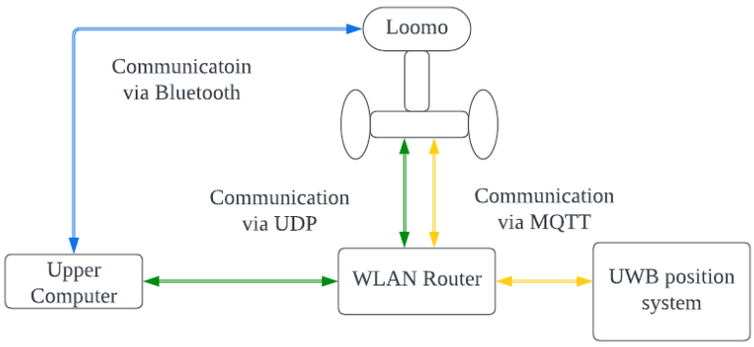

FIGURE 5.1: Communication frame of system

As shown in figure 5.1, there are three links in the whole communication framework. The blue channel of indicates the Bluetooth channel, which is a link independent of the network channel and is used for the most direct and basic data exchange between the host computer and Loomo, including the control commands issued by the host computer and the coordinate information returned by Loomo. The green channel is the channel through which the upper computer connects to Loomo through the network. Since the upper computer and Loomo have to be under one network, a router is needed in order to form a network. This channel is chosen to use UDP protocol, which is the second option for Loomo communication. It can support the transfer of larger data such as pictures, in addition to performing all the functions of the Bluetooth channel. Also so the upper computer can get Loomo camera pictures through the network channel. The yellow channel is the communication link between Loomo and the external positioning system. In addition to the network environment, it needs an MQTT server to publish the coordinate information calculated by the positioning system. When working, Loomo sends a subscription request for coordinate information to the server via the MQTT protocol, and after the connection is established, the positioning system continuously sends coordinates to Loomo. This communication structure has a redundancy system that can mitigate the effects of instrument failure caused by fire to some extent.

5.2 Loomo Application

The Loomo application is the key to the whole system,as is shown in figure 5.2, it consists of three sub-modules, the communication module, the sensor module and the motion control module. The communication module is used to communicate with the host computer and the MQTT server of the positioning system. It needs to be compatible with Bluetooth, MQTT and UDP protocols, and can switch between Bluetooth and network communication methods. The communication system is responsible for establishing communication with the host computer to listen for commands and send data. The communication module also has the function of subscribing data from the external UWB positioning system. The sensor module is responsible for operating individual sensors, including attitude sensors, odometers, speedometers, and ultrasonic sensors. In addition, the sensor module takes on the job of data integration. In implementing the positioning function, he needs to combine the data from the attitude sensor, odometer, speedometer, and external positioning system to infer the actual position of Loomo. Since the measurement interval of each sensor is different, coordinating their timing is also one of the important tasks of the sensor module. The motion control module is a class related to Loomo's chassis, and it is also the endpoint for the data flow from the first two modules. The motion control module needs to move the Loomo according to the commands obtained by the communication module combined with the data obtained by the sensors. Specifically, it implements collision prevention by means of data from ultrasonic sensors and motion control by means of fused coordinate information combined with algorithms.

5.3 Host computer

The host computer is the interface between the user and the whole system. It has a graphical user interface (GUI) and a communication module.The GUI includes a connection interface and a control panel. In the connection interface, you can select the devices that need to establish communication. The interface can support the function of switching control among multiple Loomo. The control panel

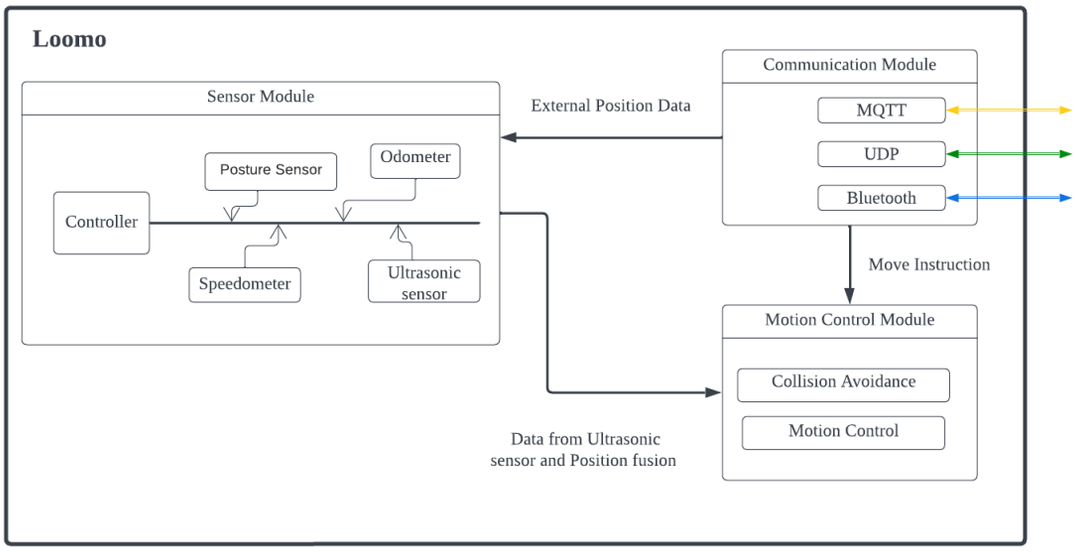

FIGURE 5.2: Block diagram of Loomo application

is used to operate Loomo, and it has two command interfaces, the command for direct movement and the command for coordinates. It also has a window to display information from Loomo, including speed and coordinates. besides, a preview window is left in the GUI to view the image information coming back from the camera. The communication module has two communication interfaces, Bluetooth communication and UDP protocol communication. The user can switch between the two types of connections.

6 IMPLEMENTATION

This section describes the implementation process of the system. Firstly, it introduces the hardware and software platform. Next, it shows the functional implementation and technical details of each subsystem. Finally it explains how the modules work together to achieve the functionality in the requirements.

6.1 Software and hardware platform

This project is developed based on Loomo robot and Android studio.The Loomo robot is a combination product of a combination of a self-balancing vehicle and a companion robot from Segway Robotics. In addition to being a vehicle and a personal robot, when set to developer mode, Loomo can install Android applications developed by individuals.

In terms of hardware, Loomo is divided into two parts, the chassis and the body. As shown in figure 6.1[13], the wheels and their connecting parts in the lower office part are the chassis. The chassis includes two wheels and their respective motors, and each wheel is equipped with an independent Hall sensor to provide stable odometry data. One is the IMU of the robot's body and head that can be used to measure the robot's pose and head orientation, and the other is the ultrasonic sensor in front of the body that can detect obstacles in real time to prevent

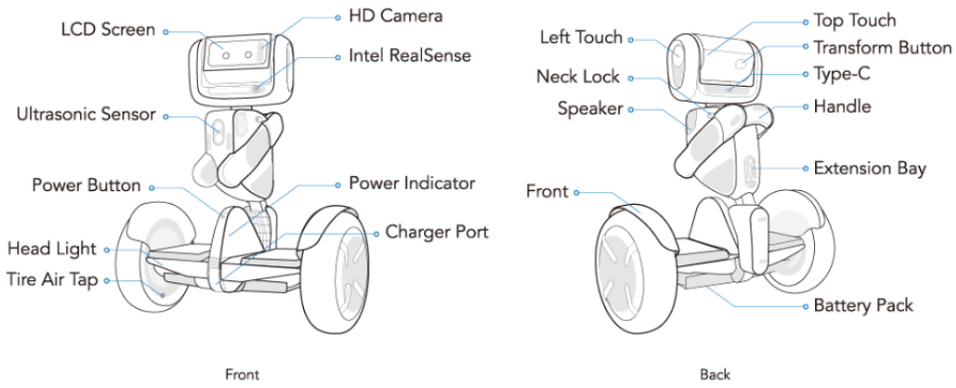

FIGURE 6.1: Loomo mechanical structure

collisions. In addition, the robot is equipped with Intel's Real Sense module, who is able to provide RGB-depth images, which provides a great help in the final attempt of the project for visual simultaneous localization and mapping (SLAM).

On the software side, the Loomo robot has a complete Android API that can be used by simply configuring the dependencies in the Gradle configuration file of the Android studio project. The API abstracts the components of Loomo, making it easy for developers to understand and operate it from the software level. However, it should be noted that the current version of Loomo is only compatible with Android SDK version for Android 5.1 and below[14].

6.2 Loomo Sub system

In Android development, if the main thread has a time-consuming operation, it will block the user interface. In fact the Android development manual does not recommend time consuming operations in the main thread and will encounter application not responding (ANR) if the main thread is blocked for more than 5 seconds[15]. In practical development, the compiler does not even allow any network operation to be included in the main thread, even if it is not time consuming. Loomo is composed of three modules: the communication module, the sensor module and the motion control module. Each module requires blocking operations, such as waiting for communication connections, waiting for sensors

to return data, etc. Therefore, each module is a separate thread, and for some complex modules even nested sub-threads are required.

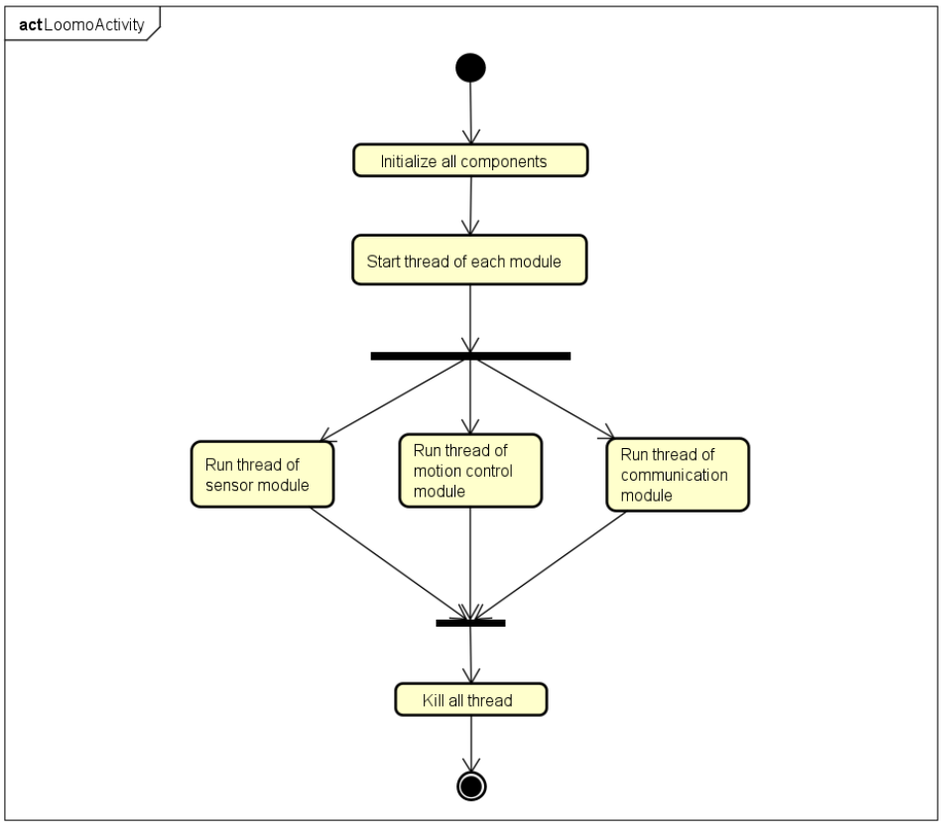

FIGURE 6.2: Loomo activity overview

During the implementation of Loomo's functions, the general flow of the program operation is shown in figure 6.2. After the Loomo application is started, the program will first enter the initialization phase, in which all classes are created and all services are registered. Then, after waiting for the initialization to complete, all threads will be started in sequence. These threads will run continuously, completing tasks independently or cooperatively to keep Loomo running properly. After Loomo completes its task, the threads will be killed artificially. When the last thread is killed, the Loomo program is finished running. The rest of this section will focus on how the various modules in the Loomo program, the three main threads in parallel, are implemented. It will also explain how the individual threads communicate with each other.

6.2.1 Communication module

In the communication module, Loomo plays the role of a lower machine or client. The purpose of its threads is to continuously listen and send large files. This module has three functions, which is communicate with other devices via three protocol:

6.2.1.1 User Datagram Protocol(UDP)

UDP is a connectionless communication method. Although it is not very reliable, sometimes UDP can be more advantageous when data needs to be received faster and small errors can be tolerated. In Android Studio, it can be used with user authorization by calling the relevant JAVA network API. In the application, a class UdpClient is used to encapsulate the communication methods. Thus UdpClient is used to communicate with the upper computer. As a thread it continuously listens to the data sent by the host computer. When a command message is received from the host computer, it is forwarded to the motion control module via Handler. Since the UDP protocol also takes on the task of sending pictures, when it needs to send packets, it opens a temporary thread to take care of the transmission so that the main thread does not block. Since the UDP protocol does not need to establish a connection, only an IP address and a port need to be recorded during initialization.

6.2.1.2 Message Queuing Telemetry Transport (MQTT)

MQTT is a lightweight communication protocol based on the publish/subscribe model [16]. As shown in figure 6.3, the model consists of three roles, subscriber, publisher and broker.The subscriber subscribes to content of interest to the broker through a data tag, which is called topic. The publisher sends the data and its corresponding topic to the broker. When the subscriber completes the subscription, the broker forwards the corresponding data to the subscriber.

In this project, the publisher is the UWB positioning system, Loomo is the subscriber, and an mqtt broker service is deployed on the server. On the subscriber

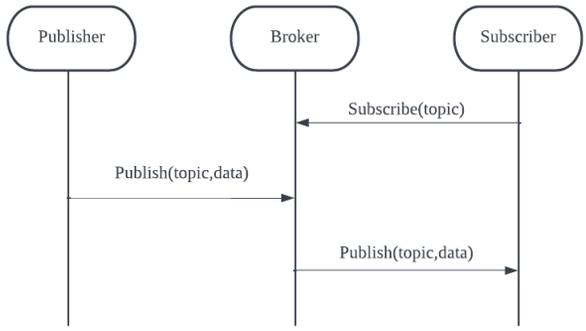

FIGURE 6.3: MQTT communication structure

side, an mqtt client is implemented to subscribe data, using the paho library. paho is a multi-language MQTT library developed by Eclipse ([17]). To use it, you need to configure two dependencies in Gradle first, as shown in figure 6.4.

```
// MQTT Service
implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.1'
```

FIGURE 6.4: Paho android studio dependency

After introducing the dependencies, a class called MQTTClient is used to encapsulate the methods of paho, as shown in figure 6.5. Paho requires the URL of the server and the subscription topic to subscribe to the data. In addition, the username and password are used for secure authentication of the connection. Note that Paho uses a callback function when receiving subscribed data, where the subscribed data and topics can be read. During coding, a Listener is used to listen to the data and send it to the sensor module.

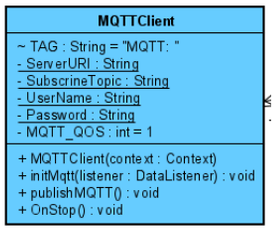

FIGURE 6.5: MQTT client class diagram

6.2.1.3 Bluetooth

The Android SDK provides a Bluetooth development kit, which can be used after applying for Bluetooth access to the device. Unlike UDP and MQTT, which can use IP addresses as the unique identifier of a device, Bluetooth is more complex to establish a connection. Although Bluetooth is a duplex communication, it needs to be divided into master and slave roles when establishing a connection. When communicating, the master needs to search for surrounding devices, while the slave needs to turn on the device search visible. Once the master finds the slave, the two are then paired based on the device name. Since Loomo acts as a slave in Bluetooth communication, its code is relatively simple compared to that of the master. Its core functional class is shown in figure 6.6.

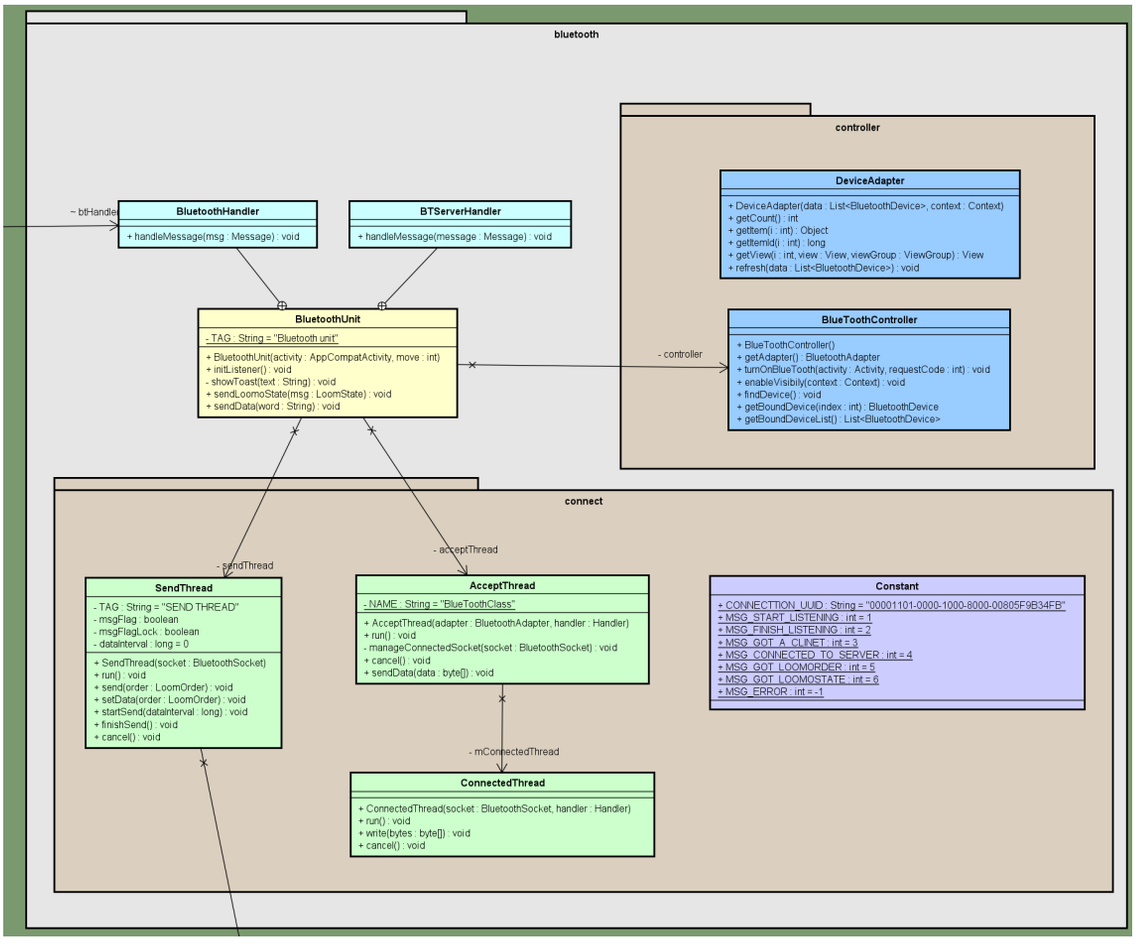

FIGURE 6.6: Bluetooth module class diagram

BluetoothUnit is the top level class of Bluetooth, it encapsulates all the members

and methods related to Bluetooth functionality. The class named BlueToothController is the hardware abstraction of Bluetooth. This class is used to operate the Bluetooth module from the hardware level, such as switching Bluetooth on and off, making it visible to surrounding devices, etc. It is called during the initialization of BluetoothUnit. ConnectedThread is the abstraction of a single connection as a slave, each instance represents a communication that has been established. This is a class that is used to continuously listen for messages from the host. Given that Bluetooth is a duplex communication, this class also has the function of sending messages to the host. Because of the small amount of data in Bluetooth communication, there is no extra thread to send data. AcceptThread is a collection of abstractions of Bluetooth as a slave, that is, it is a collection of ConnectedThread. In the work, you can connect many different devices through Bluetooth, for each connected device, there will be a ConnectedThread to communicate. AcceptedThread is responsible for creating, disconnecting and scheduling these connections. In addition, this class is responsible for message forwarding, it will forward the messages obtained from external devices via Bluetooth to the corresponding internal module. For the moment, only one Loomo communicates with a single desktop, so AcceptedThread exists more as a wrapper for the connection. However, when multiple devices are involved in the communication, this class ensures the extensibility of the communicating devices.

6.2.2 Loomo sensor module

In the sensor module, data needs to be read from the ultrasonic sensor and the IMU. the data from the IMU, when acquired, is fused with the data from the external positioning system obtained by the communication module. Finally the data from both are passed into the motion module.

6.2.2.1 Ultrasonic Sensor

The ultrasonic sensor used here is the one built into Loomo. It is capable of detecting the distance of obstacles in front of the robot, it has a detection distance of 250 millimeters to 1500 millimeters and a detection angle of 40 degrees. The Loomo SDK provides the appropriate API to call it, and only requires adding dependencies in the Gradle configuration file. However, before using any Loomo SDK, it is necessary to connect to the service. This part is placed in the initialization phase of the program(figure 6.2). In Loomo SDK, all sensors are encapsulated in the Sensor class, which is a single-instance script that can be used to fetch the ultrasonic sensor detection data through its instance anywhere in the program after the service has been registered.

6.2.2.2 Inertia measurement unit, IMU

The IMU-based odometer is also included in the Loomo SDK. It provides an estimate of the robot pose (x, y, direction) relative to the starting pose. As shown in figure 6.7, according to the manual [18], the origin of the coordinate system is the center of the chassis. The x-axis points to the front of the robot, the y-axis points to the left, and the z-axis is perpendicular to the ground. The angle of rotation (theta) follows the right-hand rule. It is updated every 50 milliseconds.

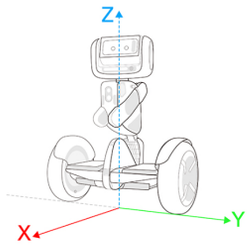

FIGURE 6.7: Odometry reference plane

In the program, a thread called Odometry is the abstraction of the odometer. As shown in 6.8, it works by first reading data from the odometer, then adding it to the offset, then storing it in the data cache and setting the data availability flag. Finally the thread hangs for a certain interval.

- The odometer reading is a value relative to the starting point.
- The odometer data is refreshed every 50 milliseconds.

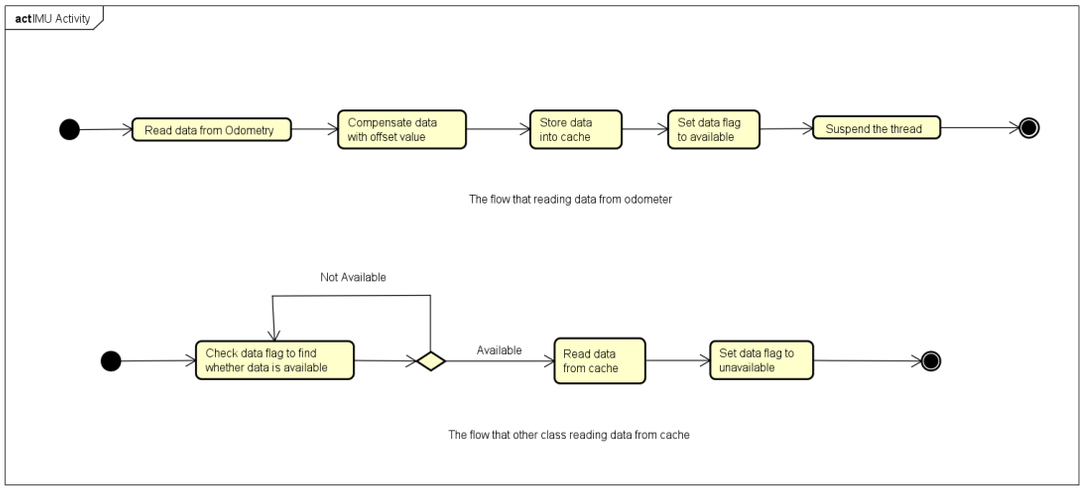

FIGURE 6.8: IMU activity diagram

For the first feature, the offset variables for the odometer readings are reserved in the class. Further, the Odometry class acts as a thread whose main purpose is to solve the second problem, which is to request the odometer results at regular intervals and subsequently to perform a simple processing of the results. The minimum time interval for reading data is set to 50 milliseconds seconds. Although the time consuming data processing in the 50 ms scale is not negligible, it is still not recommended to set the reading interval smaller. Because the point that needs attention is not to improve performance but to prevent misreadings. Such a high frequency of feedback at low speed is already enough for the robot to adjust its own motion. Due to the timing constraints of this thread, a data cache is introduced in order to be able to couple with other classes. Its role is to temporarily store data that is already available so that it can be read once and only once. Specifically, access is controlled by a data identifier that is marked as unavailable when the cache has been accessed, but is reset when new data is deposited.

6.2.3 Loomo Motion control module

The motion control module is an abstraction of Loomo's chassis, and the hardware associated with it are the motors and the Hall sensors on the wheels. The original API provides two modes of motion. One is to use the coordinate system, which is on a temporary Cartesian plane with Loomo as the origin, Loomo's direction as the positive Y-axis and the right direction as the positive X-axis. By this method, it is easier to move Loomo to a specified position. Another method is to set the speed on its wheel motors separately. This is a more flexible and maneuverable mode by which loomo can easily perform tasks such as pivot steering. While each mode has its own advantages, they share a common disadvantage in that they lack feedback control, i.e., both modes are open-loop. The open-loop control can still be used when a person is available to operate it. If Loomo is needed to go somewhere on its own, it will definitely get lost. So an improved motion control algorithm was needed.

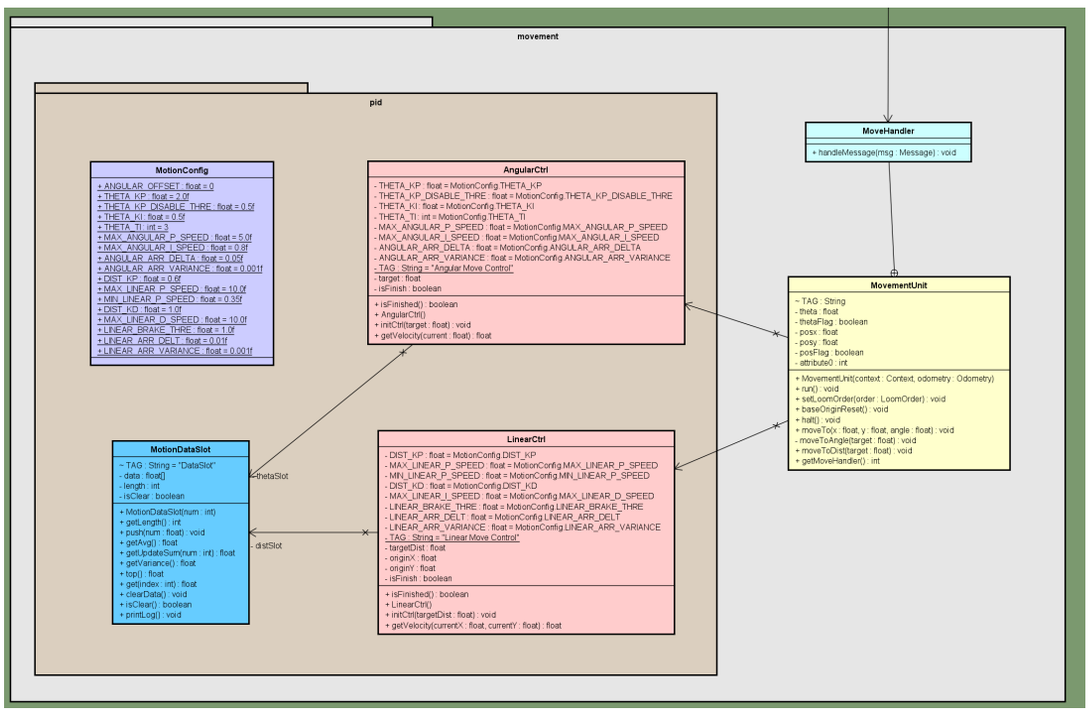

FIGURE 6.9: Motion control class diagram

The motion control module was thus designed. As shown in figure 6.9, MovementUnit is the top level class which encapsulates all the methods related to sensors and motors. It is also a thread that listens to the upper computer commands

forwarded from the Bluetooth side and makes the corresponding operations accordingly. There are two modes of operation, a direct control by purely calling the API, and a coordinate mode based on direct control and sensor feedback. In the coordinate mode, Loomo is able to move in polar coordinates. Its movement process is divided into two steps, the first is to turn to the corresponding angle first, and the second is to move forward some distance. In each of these two movement stages, PID control is used. The PID algorithms are implemented in AngularCtrl and LinearCtrl respectively. PI algorithm is used in AngularCtrl and PD algorithm is used in LinearCtrl.

6.2.3.1 Linear velocity control

The PI control algorithm is a control algorithm used to eliminate static differences. The static difference in this case means that Loomo will oscillate near the target point. There are two reasons for this phenomenon. One is that Loomo is a balance car, which needs to be fine-tuned back and forth to stay upright when it is stationary, and therefore deviates from the target point. The second reason is that the machine is objectively in error and cannot stop precisely at the target point. These aforementioned small errors will make the proportional element work. On the other hand, the coefficients of the proportional element cannot be too small in order to ensure the response speed, which causes the effect of the proportional element to be over-tuned in small errors, thus causing oscillations. In summary, the integral element is introduced to gas to eliminate the oscillation. When Loomo is close to the target point, the proportional element is deactivated and the integral element plays the main role to achieve a non-linear slight adjustment effect by integrating the small errors.

The flow of the linear velocity control is shown in figure 6.10. First, a target point needs to be set and the distance to the current target position of the robot is calculated, which is the error. The error is then stored in the data cache and is reserved for stability judgments. The cache is a queue to store a certain number of data. Given the interval of reading position information is fixed, the cache is actually store data of a period of time. Next, the error is used to determine whether the braking is required, and the threshold for braking is an empirical constant.

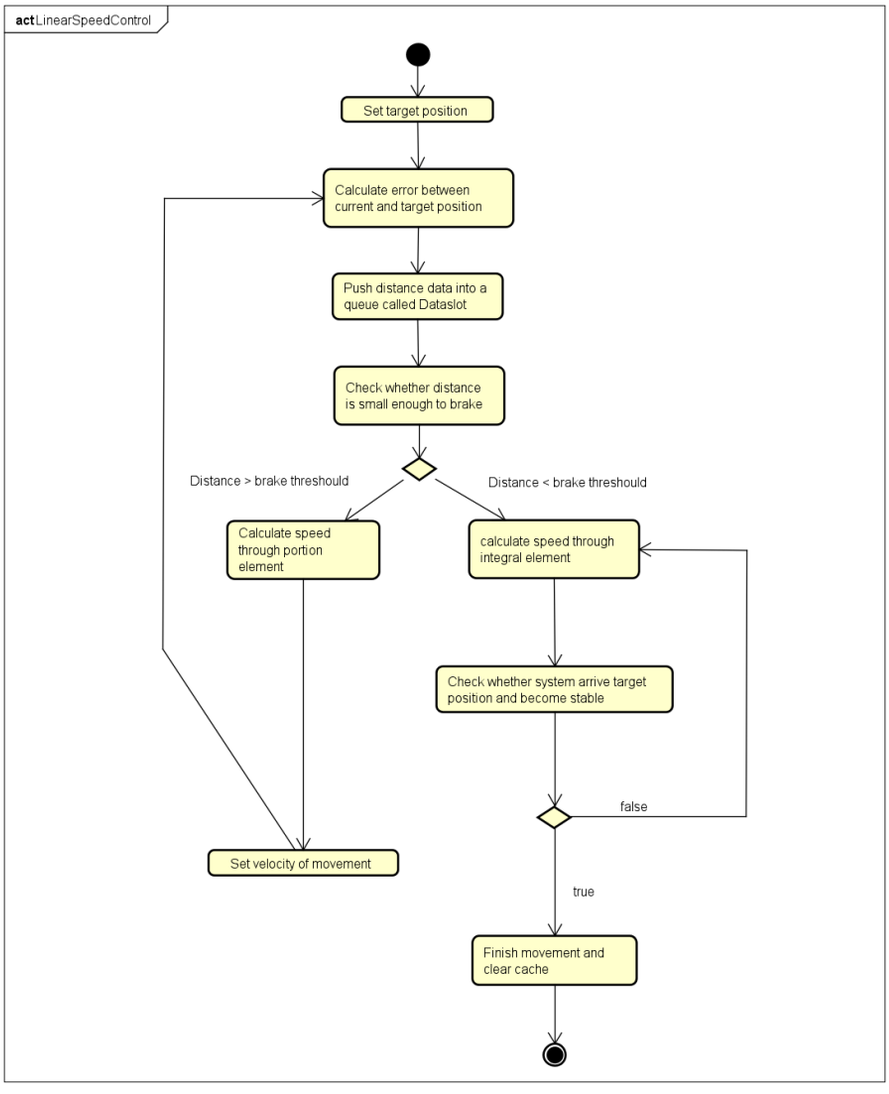

FIGURE 6.10: Linear velocity control activity diagram

If the error is greater than the braking threshold, then a proportional element is used as the mainstay in the speed control. The error is multiplied by a parameter that limits its upper and lower limits, and the result of the session is subsequently used as an input to the speed. This parameter is called the proportionality factor.If the error value is less than the braking threshold, then the braking phase will be entered. In the braking phase, the proportionality factor is deactivated by setting it to a very small value. At this point, the integration element plays a major role in the control.The integration session integrates the historical errors and multiplies them by a factor that is set as an input to the

velocity. There are two key parameters in the integration session, namely the integration time and the integration factor. The integration time refers to the time period in which the historical error involved in the operation is located, and the integration coefficient is the coefficient multiplied after integration.After the integration factor, a stability judgment is performed. In the stability judgment, the critical stability judgment is performed first, and then the stability judgment is performed. When judging critical stability, the average value of the cache data is found. If it converges to zero, then the system is critically stable. At this time, there are two situations, one is that the robot has reached the target position and the speed is zero, and the other is that the robot is swaying back and forth near the target position. Therefore, a second determination is needed. The second determination is made by taking the standard deviation of the data to determine whether the robot is stationary or not. If it is stationary, then the robot arrives at the destination and remains stable.

6.2.3.2 Angular velocity control

Figure 6.11 shows the flow chart of angular velocity control. The angular velocity control adds an angular mapping and differential element to the control of the line velocity. Since the output range of the angular sensor is from 0 to 360 degree, this means that the angle can change abruptly at a point of the circumference of the circle. This abrupt change can cause the system to oscillate in dispersion and then collapse. Therefore, to ensure that an accurate error value can be obtained, the current angle is mapped between -360 degree and 4 degree, and then the error is calculated. In addition, the direction of rotation needs to be calculated to ensure that the robot rotates through an inferior arc.

The initial angular velocity control did not introduce a differential element. The consequence of this was that the robot started extremely slowly and took a long time to reach critical stability. To this end, an attempt was made to increase the parameters of the proportional element, but this in turn caused equal amplitude oscillations beyond the critical point. Finally, a differential element was introduced for override control. The principle of the differential element is to advance the control by predicting the motion trend. In the current case, the error value is

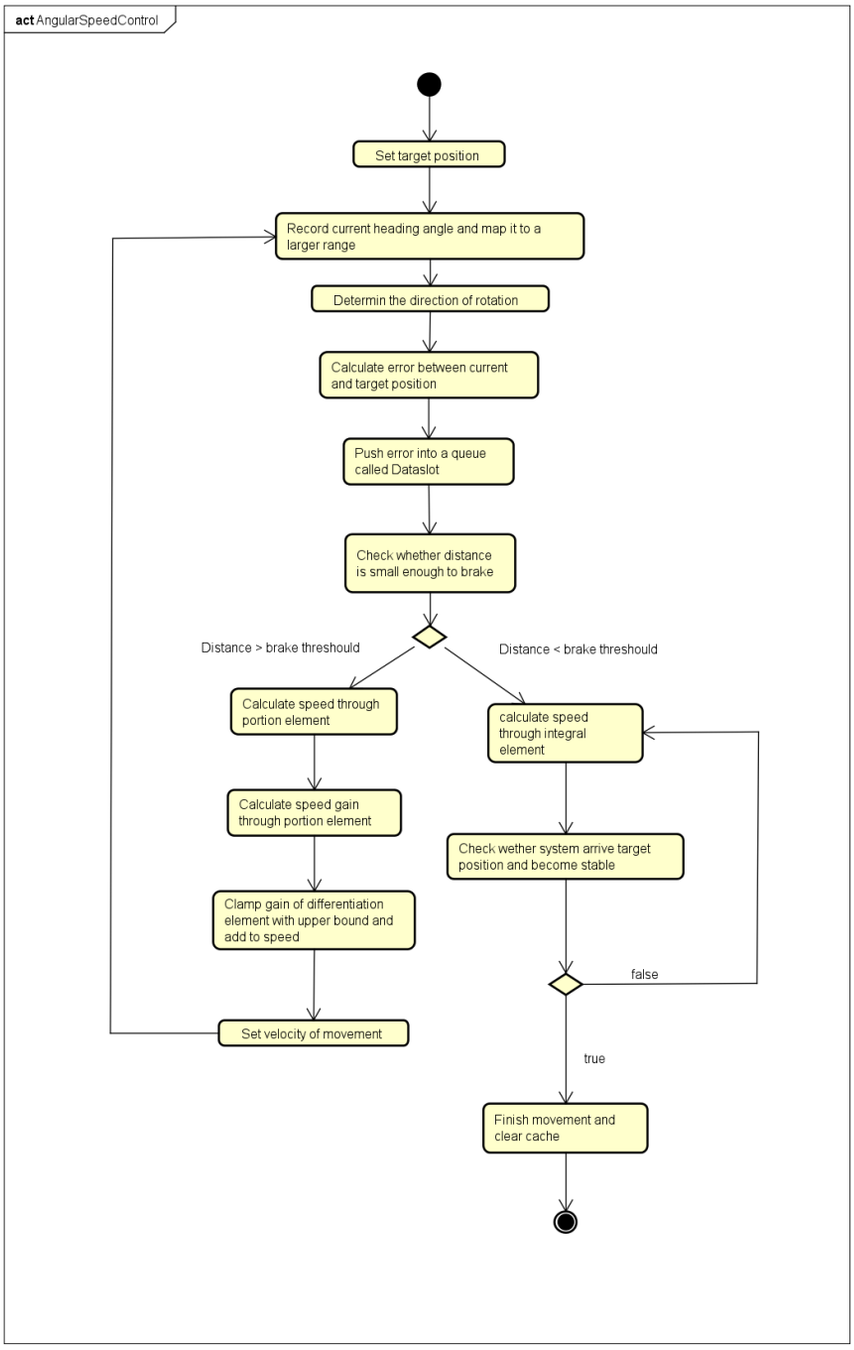

FIGURE 6.11: Angular velocity control activity diagram

positively correlated with the velocity after the differencing is performed. During the start-up phase, the rate increases and the differential element effect is significant, so that a larger start-up speed is obtained. As the target position is approached, the effect of proportional integration diminishes, so the rate decreases and the differential element effect is cut, further reducing the speed.

6.2.4 Host computer

The OP is an Android application which can be deployed on a tablet or a cell phone. The OP plays the role of a user interface in the system, through which Loomo can be operated. Communication is the most important function of the upper computer. It is required to be able to connect with Loomo using both UDP and Bluetooth.

There is no role of master and slave for UDP communication, it is peer-to-peer. The implementation of the UDP function on the upper machine is generally the same as on Loomo, so it requires only a simple porting of the code.

Bluetooth communication, on the other hand, is relatively more complex. In Bluetooth communication, the host computer takes the role of the host. Before communication, it is necessary for the host to search for nearby Bluetooth devices, find Loomo, and initiate a pairing request. After a successful pairing, a communication connection request needs to be initiated by the host computer. Only when the connection is established, data can be sent and received. Since the host computer sends small data such as commands and receives large data that may contain pictures, the host computer opens threads for receiving, but not for sending.

6.3 Cooperative working of modules

6.3.1 Collision Prevention

This part describe how to use ultrasonic sensor to implement Collision prevention function. The ultrasonic sensor can detect obstacles from 250 mm to 1500 mm in a 40 degree range ahead. A threshold value between ranges is set to serve as the minimum distance from Loomo to the obstacle. When Loomo is less than the minimum threshold, the Loomo speed is set to 0. At this point, if a non-backward command is received again, the command is simply skipped and the Loomo is left in standby in place. It will also work in coordinate mode movement. In coordinate movement mode, Loomo will adjust the angle first before

moving forward. After the angle is adjusted, a determination will be made and if the distance to the obstacle is less than the sum of the forward distance and the minimum threshold, then no movement will be made.

6.3.2 Fusion Positioning

When it comes to navigation, it is not possible to use electromagnetic position only because of its low response speed. However, it has relatively high accuracy. Besides, the in-build IMU sensor has high data frequency but low accuracy. The two sensors can complement each other. This part will introduce how to make a usable navigation system in combination of this two sensor.

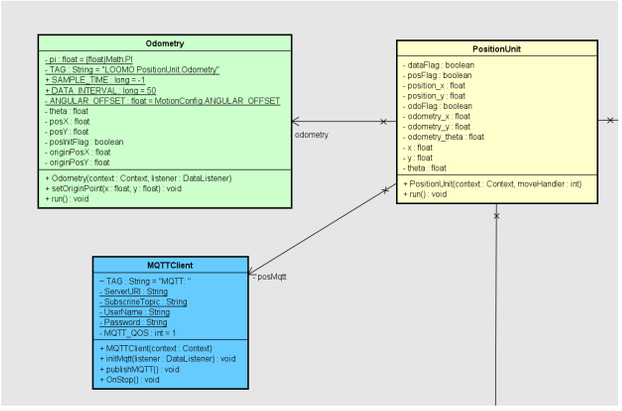

FIGURE 6.12: Class diagram related to fusion positioning function

As shown in figure 6.12, the class named PositionUnit is the top-level class of the location system through which other classes get location information. It can access IMU data through a class named Odometry and also has a Message Handler to receive location data from MQTTClient. When PositionUnit receives the external location information, it will update its own location information immediately. At the same time, it will immediately read the latest IMU data and calculate the IMU compensation by comparing the two data, with the positioning system data prevailing. The compensation value is then updated in the Odometry class. In this way, until the next time the positioning system data is obtained, the IMU data will be based on the current data.

7 EVALUATION

7.1 Evaluation

Previous chapters described how to call Loomo's sensors and fuse them into Loomo's control. This chapter will evaluate the effectiveness of motion control with multiple sensors involved by comparing it to the sensorless or single-sensor control methods of original API.

7.1.1 Direct Manipulate

In the direct manipulation of Loomo, Bluetooth is used as the communication medium to transmit the mobile commands from the upper computer in real time.

In terms of communication quality, the Bluetooth manipulation performs better at a short distance indoors. All operations are responded to in real time. At longer distances, about 20 meters away, the Bluetooth signal starts to become unstable and produces a delay in response. When the distance increases further, there may be dropouts or Loomo end application crashes. By reviewing the crash log, a data timeout error was reported. When using Bluetooth communication, the upper computer receives data stably, in real time and without errors. It is

able to receive coordinate information. Again, when the distance is long, there is a delay in reception or dropped connection.

When using Bluetooth to control the robot to move directly can complete the basic functions, including forward, backward, left and right steering. However, in the event of a delay, Loomo will mount up directly if an obstacle is encountered in front of it. Ultrasonic sensors can participate in the function of obstacle avoidance can improve the situation. When the front release detects an obstacle, it is able to stop and will not respond to forward and steering commands. However, the backward command is not affected. When Loomo back away from a sufficient distance to be able to continue to operate normally. This effectively realizes the function of anti-misoperation. And in the robot manipulation sometimes will lose the field of view, when the positioning system to provide the sensor coordinates to a certain extent to allow the operator to grasp the position information.

7.1.2 Move by Coordinate

The movement by coordinates is mainly used for path planning. In the coordinate movement mode, the participation of multiple sensors plays an important role. the original control command in API only has Hall sensor's as feedback, which can accurately reflect the number of laps of wheel movement, but the inference of movement distance only by the number of laps will cause a large error. In this project improved movement control method has UWB positioning system, IMU and Hall sensor together, so that its performance has been greatly improved.

In terms of rotation performance, the original Loomo self-rotation method cannot even complete accurate steering on flat ground, and when the rotation encounters resistance, the final orientation angle will be greatly deviated.PID angle control allows the robot to reach the target angle faster and more accurately, and fine-tune the correction at the target orientation. When the rotation encounters resistance, such as running over a book, the final orientation is not affected. When there is a slight obstruction in the rotation, Loomo will continue to turn to the target position after the obstruction disappears.

In terms of straight-line movement, PD control performs equally well. After giving a command, the robot starts quickly and responds quickly, decelerates in time before reaching the end point, and finally can reach the target point by fine-tuning. In addition, the straight-line movement command is also included in the API, and it is fairly accurate in terms of distance. However, if it encounters a damping, such as a slope, or an obstacle, such as a stone or a book, then the robot will stop before reaching the target point. It can be presumed that the original method of movement is through the wheel hitch in the Hall sensor as feedback. The improved version of the control method uses inertial sensors as feedback, which are not affected by road conditions, and therefore can compensate for these problems.

8

FURTHER WORK

Beyond the requirements, there are more functions that can greatly contribute to fire rescue robot. Further, some other sensors on Loomo remain to be discovered, such as real sense camera, microphone array, etc. Actually some work also has been conducted to explore more possibility of Loomo and its sensors. In the final stage of this project, related work on SLAM was attempted.

8.1 Overview

SLAM is a method of reconstructing maps and localization from images, which identifies landmarks in the environment and builds maps by recording the current position of the observer relative to the landmarks. Also, it can locate landmarks by comparing their relative positions with those of previously completed maps. In this project, we try the image reconstruction method combining existing sensors, in which the completed links are image acquisition, image filtering, feature point extraction and point cloud map spatial mapping. The first two links are executed on Loomo, and the last two links are executed on the PC side. The first two links require the acquisition of several RGB-D images in a space and the corresponding external camera parameters. In the latter two sessions, feature points in these images need to be extracted by the orb algorithm and the RGB pixels corresponding to the feature points need to be projected into the

space to reconstruct the scene by using the camera external reference and the depth map.

8.2 PC Environment

Ubuntu is used as the operating system on the PC, the advantage of Ubuntu is that it has a powerful package management system to resolve package dependencies and Ubuntu has good community support. In practice, ubuntu 16.04 was chosen as the operating system. The programming language of choice was c++, and cmake was chosen as the compiler. C++ was chosen for its speed advantage and opencv library support. cmake was chosen because of its good support for lower OS versions. In addition to Opencv, the library named pcl was imported for point cloud visualization and eigen3 was imported as a matrix manipulation tool. With the help of android ndk, all the c++ code can be ported to Loomo eventually.

8.3 Data Acquisition and Filtering

The purpose of this part of the work is to capture data from Loomo and convert it into a format that can be processed by the algorithms on the PC side. The figure shows the structure of the classes associated with it on Loomo. The camera is called in the RealSenseCamera class. Figure 8.1 is written to the storage in ImageIO and the filtering of the image is performed in FrameFilter.

8.3.1 Acquisition

In this session, each piece of data to be acquired consists of an RGB image, the corresponding depth map and the camera extrinsic reference at the time of capture. The external camera parameters refer to the pose of the camera when the image was captured, namely spatial coordinates and spatial orientation.

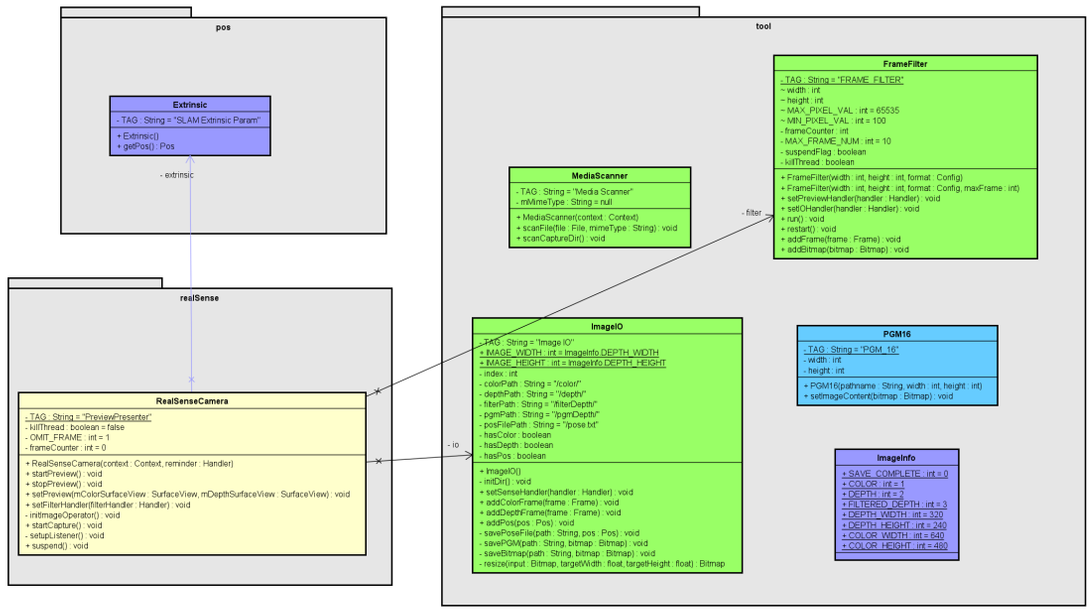

FIGURE 8.1: Loomo slam class diagram

The class named RealSenseCamera is the top-level class and the abstraction of RealSense Camera. In this class, we use the library provided by the Loomo SDK to call the camera for image acquisition, and like other components of Loomo's, we only need to register the service to call it. pixels, 320 by 240. So it is enough to compress the RGB image. In terms of format, the RGB image needs to be saved as a png, which can be done using the package that comes with Android. The depth map needs to be saved as PGM, a grayscale storage format, which is wrapped in a class called PGM16.[19]

8.3.2 Filtering

During depth map acquisition, the camera actively emits infrared light, which returns to the receiver when encountering an object. Then a image was derived according to the reflected light. However, it is a problem that when infrared beams are cast to an object, those with curve surface in particular, some of them are reflect to other place, and thus lead to voids appear on the image. For example, as the figure 8.2 shows, actually there is a complete wall on the left of chair, but it looks full of holes.

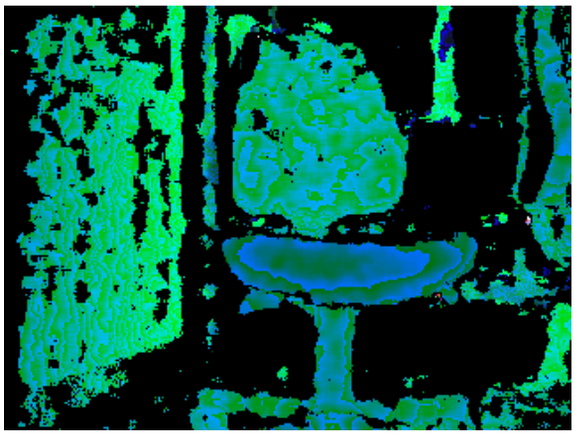

FIGURE 8.2: Voids on origin images

In practice, finding and filling a hole is a time-consuming operation, so a new thread is opened to do it. A class called FrameFilter exists for this purpose. It has a cache for keyframes and a buffer for images over a period of time, called trivial frames. When the thread runs, the key frames are compared with the trivial frames. If there are holes in the key frames but not in the trivial frames, the holes are filled. The optimization ends until every trivial frame is traversed. The result of optimized is shown in figure 8.3.

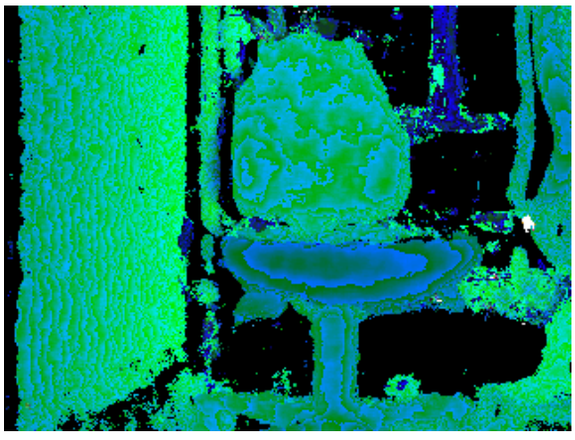

FIGURE 8.3: Some voids are eliminated through filter

What's more, key frame is useful in synchronization.The RGB image and the out-of-camera reference are taken from one moment in time, while the depth image is taken from a period of time, so it is necessary to make all three consistent. Since keyframes are used, the RGB image and depth map keyframes can be synchronized as long as they are guaranteed to be recorded at the same moment when the image is acquired.

8.4 Scene Reconstruction

8.4.1 Feature Point Extraction

Several frames of information are required for scene reconstruction. The feature points are extracted between frames by ORB method. ORB algorithm forms a matrix of a pixel point and all the pixel points around it with radius 2, and determines whether a point in two maps is the same point in space by comparing the similarity of each matrix feature value in the two maps.[20]. Further, when reconstructing scene, only feature points need to be take into consideration, so the ORB algorithm not only links images taken at different locations at different times, but also greatly reduces the number of points to be projected, thus improving the efficiency of spatial mapping.

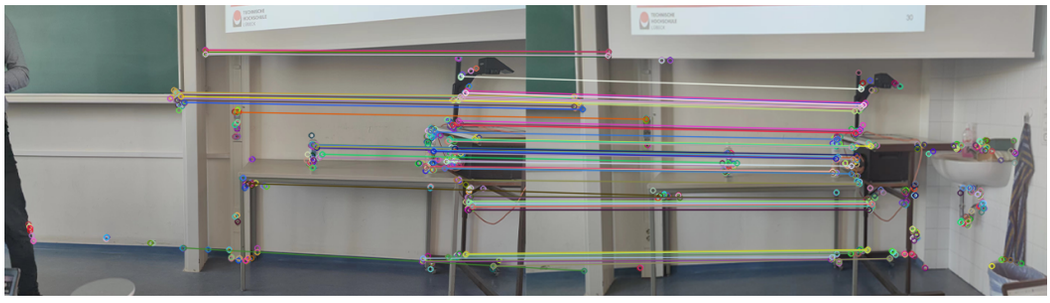

FIGURE 8.4: Example of point match in two picture

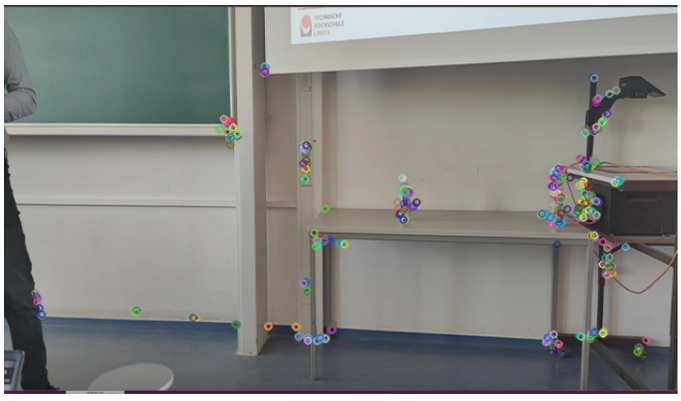

FIGURE 8.5: Result of feature point extraction

8.4.2 Spatial Point Projection

In this step, camera extrinsic parameter, RGB image and depth image are combined to reconstruct the scene. Through the camera extrinsic parameter and depth image, the spatial position of each pixel in depth image can be calculated. Finally, it just need to find the corresponding pixel points in the RGB image and project them into space to reconstruct the scene.

8.5 Discussion

Although some part of SLAM has been tested. There are still many necessary aspects of SLAM that have not been implemented, such as loopback detection, object recognition, context determination, etc. Therefore, the current progress is far from being able to exploit the usefulness of SLAM. However, this project has some scalability and provides an interface for possible subsequent attempts to carry SLAM functions.

9 CONCLUSION

9.0.1 Conclusion

In this project, the effects and possibilities of multi-sensor fusion are tried in the direction of a fire-fighting robot. Ultrasonic sensors are introduced as collision detection in the manipulation aspect to increase operational fault tolerance. For path planning, internal and external sensors are used to enhance localization and response time for better motion control. In the application of SLAM, inertial measurement units are used to replace the role of visual odometry, which greatly reduces the complexity of the algorithm.

On the other hand, there is still a lot of work to be done. For example, the re-connection of Bluetooth devices and the automatic switching of communication methods still need to be improved. The robot's self-path planning has also not yet been deployed, and currently requires human control.The functionality of SLAM has not yet taken shape and needs further development and research.

Even so, in this project, multi-sensor fusion has proven itself to be effective and promising in all aspects. If further improvements and research are carried out, an excellent fire rescue system can be completed on this foundation.

9.1 Outlook

In future work the efforts on these aspects can make the system more complete and high performance, which is path planning, communication and SLAM.

First, a more robust communication system needs to be put in place, not only at the software level but also in the hardware. Ideally, a network with a strong signal, high bandwidth and low latency can cover the affected area, such as a 5G network base station. On this basis, the camera streaming function on the disaster relief robot can be implemented, which, together with the direct control mode, can allow firefighters to have a fire-resistant and heat-resistant machine double in the fire.

Second, a path planning algorithm needs to be built into the robot so that the robot has the ability to act on its own. So that when the firefighter is not able to manipulate it because of the loss of signal, the robot can return to the safe area or enough to go to the target location by itself.

Finally, the application of SLAM algorithms, which allow robots to autonomously perceive and explore their environment, is also worth investigating. Combined with path planning algorithms, unmanned autonomous search and rescue robots can be developed, which is very promising. It could search and rescue itself in a fire scene and send the location of the injured to firefighters. Its unmanned nature means that it can be deployed in multiple units to increase search and rescue efficiency. In addition, according to Loomo's product positioning, its role as an accompanying robot means that it may be on the scene when a fire breaks out, at which point the robot can play the role of an evacuation guide, evacuating in time to reduce losses.

REFERENCES

[1] H. Amano, "Present status and problems of fire fighting robots," in *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, vol. 2, 2002, 880–885 vol.2. DOI: 10.1109/SICE.2002.1195276.

[2] R. Murphy, "Trial by fire [rescue robots]," *IEEE Robotics Automation Magazine*, vol. 11, no. 3, pp. 50–61, 2004. DOI: 10.1109/MRA.2004.1337826.

[3] Y.-D. Kim, Y.-G. Kim, S.-H. Lee, J.-H. Kang, and J. An, "Portable fire evacuation guide robot system," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2789–2794. DOI: 10.1109/IROS.2009.5353970.

[4] J. Zhao, J. Gao, F. Zhao, and Y. Liu, "A search-and-rescue robot system for remotely sensing the underground coal mine environment," *Sensors*, vol. 17, no. 10, p. 2426, 2017.

[5] G Arun Francis, M Arulselvan, P Elangkumaran, S Keerthivarman, and J Vijaya Kumar, "Object detection using ultrasonic sensor," *Int. J. Innov. Technol. Explor. Eng*, vol. 8, pp. 207–209, 2020.

[6] L. Koval, J. Vaňuš, and P. Bilík, "Distance measuring by ultrasonic sensor," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 153–158, 2016, 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.12.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316326623.

[7] P. Zhang, J. Gu, E. Milios, and P. Huynh, "Navigation with imu/gps/digital compass with unscented kalman filter," in *IEEE International Conference Mechatronics and Automation, 2005*, vol. 3, 2005, 1497–1502 Vol. 3. DOI: 10.1109/ICMA.2005.1626777.

[8] Y. Zhang, Y. Fei, L. Xu, and G. Sun, "Micro-imu-based motion tracking system for virtual training," in *2015 34th Chinese Control Conference (CCC)*, 2015, pp. 7753–7758. DOI: 10.1109/ChiCC.2015.7260871.

[9] N. D. with Us, *Uwb minutes: Ranging technics*, https://www.youtube.com/watch?v=5KN4dJdkHUk, Last accessed on 11-6-2022.

[10] N. Mehta, D. Chauhan, S. Patel, and S Mistry, "Design of hmi based on pid control of temperature," *International Journal of Engineering Research and*, vol. 6, p. 05, 2017.

[11] A. R. Khairuddin, M. S. Talib, and H. Haron, "Review on simultaneous localization and mapping (slam)," in *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 85–90. DOI: 10.1109/ICCSCE.2015.7482163.

[12] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[13] *Loomo sdk, Overview*, https://developer.segwayrobotics.com/developer/documents/segway-robot-overview.html, Last accessed on 11-6-2022.

[14] *Loomo sdk, Setup Development Environment*, https://developer.segwayrobotics.com/developer/documents/setup-developing-environment.html, Last accessed on 11-6-2022.

[15] *Processes and threads overview*, https://developer.android.com/guide/components/processes-and-threads, Last accessed on 11-6-2022.

[16] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, 2008, pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519.

[17] *Paho*, https://www.eclipse.org/paho/, Last accessed on 11-6-2022.

[18] *Loomo sdk, Loomo SDK*, https://developer.segwayrobotics.com/developer/documents/segway-robots-sdk.html, Last accessed on 11-6-2022.

[19] *Netpbmfile formats*, https://en.wikipedia.org/wiki/Netpbm#File_formats, Last accessed on 11-6-2022.

[20] X. Gao, T. Zhang, Y. Liu, and Q. Yan, "14 lectures on visual slam: From theory to practice," *Publishing House of Electronics Industry, Beijing*, 2017.