Abstract

Based on the task of $ROBOCUP2019 - Stage1\ Farewell$, this article mainly introduces how to implement a service robot that can complete the task of picking up and dropping guests.We will introduce the overall framework of our design, the robot function platform we designed, our contribution to the software development of service robots, and the contribution of our development to related technologies and sciences.Among them, we mainly implement the loop state control system based on finite state machine, the scheme of establishing map based on SLAM and optimized EKF algorithm, the blind navigation system based on visual control and the speech recognition system based on $Keda\ Xunfei$ API.In the navigation aspect, we also designed a solution for automatic real-time pose output, known area navigation and blind navigation smooth switching. In terms of speech, we designed a speech recognition intelligent control system to realize automatic opening and closing of speech recognition. And try to improve the accuracy of speech recognition through voice filtering and other operations.Finally, we summarize our research and propose directions for improvement and further research.

Keywords: ***EKF SLAM Algorithm***; ***Speech Recognition***; ***Blind Navigation***; ***Loopback FSM***

1 *Introduction*

Our group mainly chooses to complete the *Farewell* task, and now we will further explain what we have done and what we expected to realize.

The background of the task is as follows:

Some guests are tired, so they call the robot to retrieve their coat. It's raining outside and there is only one umbrella, so the robot takes the guests one by one to their cab and returns with the umbrella.

The main goal of the task are as follows:

The robot escorts two tired guests to their cab (where the cab driver is standing).

In order to achieve this task, we break down the task into four subtasks, and finally we further decompose the task. Finally, all tasks are divided into the following stages:

1. Waiting to start and identify the customer stage
2. Picking stage and passing item stage
3. Accompanying customers to go out and find drivers
4. Recycling umbrella and navigation regression stage

Then we further split the task process, and the corresponding main technologies are given by the following stages:

Stage 1: Visual recognition and voice communication.

Stage 2: Autonomous navigation, robotic arm grabbing, visual recognition.

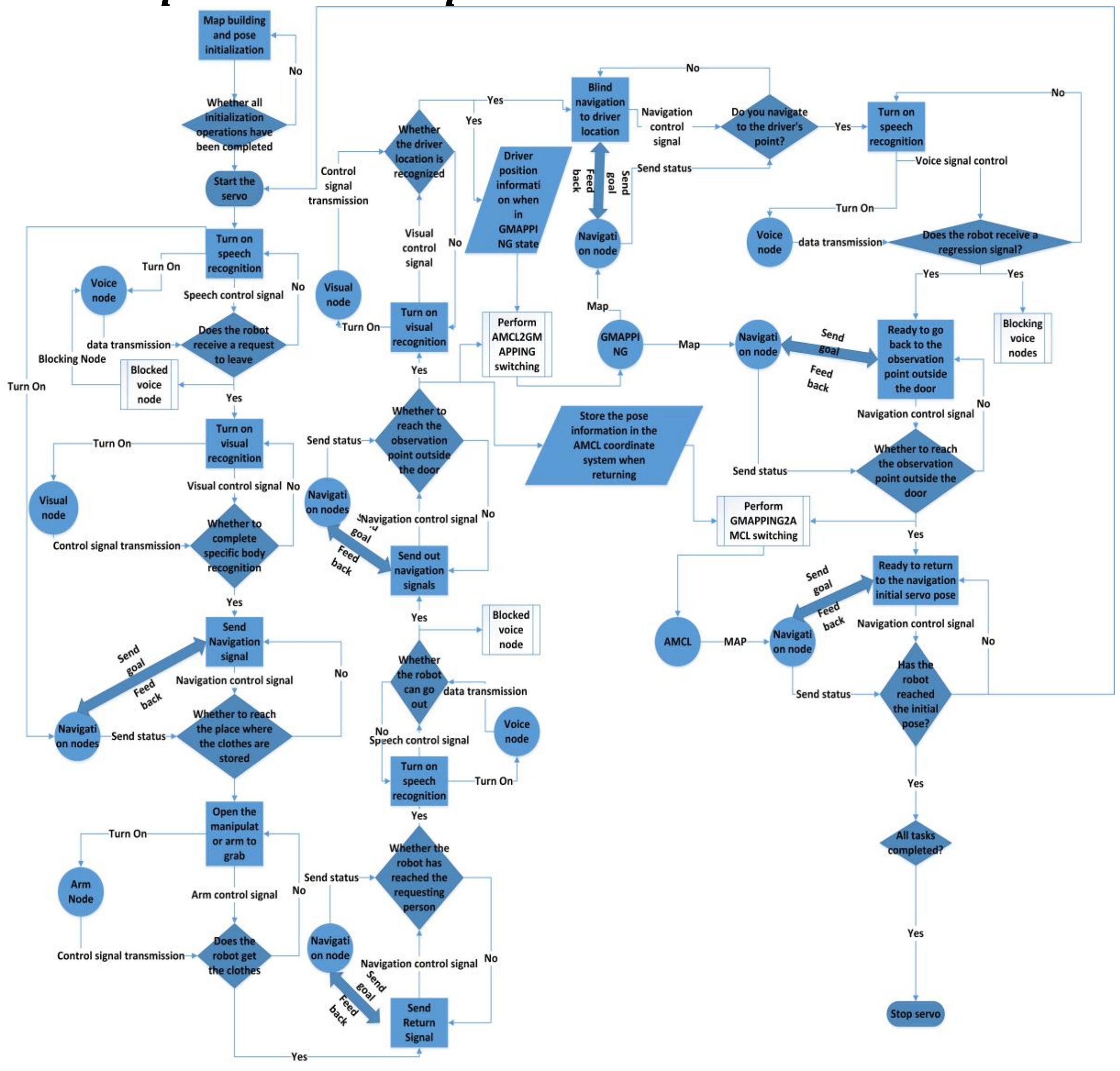
Stage 3: Autonomous navigation, blind navigation, robotic man-machine combination, visual recognition, speech recognition.

Stage 4: Robotic arm grabbing, autonomous navigation, speech recognition.

In the end, we combined the various innovative algorithms with hardware framework design and software control system design to finally complete the corresponding tasks on *turtlebot*2. Below we will introduce our research and innovation in accomplishing the above tasks in the order of hardware design, software contribution and scientific and technological contributions.

2 *Framework*

Task implementation process



Map building and pose initialization
No
Whether all initialization operations have been completed
Start the servo
Turn on speech recognition
Turn On
Voice node
Speech control signal
No
data transmission
Does the robot receive a request to leave
Blocking Node
Blocked voice node
Yes
Turn On
Turn on visual recognition
Turn On
Visual node
Visual control signal
No
Control signal transmission
Whether to complete specific body recognition
Yes
Send Navigation signal
Send goal
Feed back
Navigation control signal
No
Navigati on nodes
Send status
Whether to reach the place where the clothes are stored
Open the manipulator arm to grab
Turn On
Arm Node
Arm control signal
No
Control signal transmission
Does the robot get the clothes
Yes
Control signal transmission
Whether the driver location is recognized
Yes
Visual node
Visual control signal
No
Turn On
Turn on visual recognition
Yes
Driver position information when in GMAPPING state
Perform AMCL2GMAPPING switching
Whether to reach the observation point outside the door
Send status
Navigati on nodes
Navigation control signal
No
Send goal
Feed back
Send out navigation signals
Yes
Blocked voice node
Whether the robot can go out
data transmission
Voice node
No
Speech control signal
Turn on speech recognition
Turn On
Yes
Whether the robot has reached the requesting person
Send status
Navigati on node
Navigation control signal
No
Send goal
Feed back
Send Return Signal
Store the pose information in the AMCL coordinate system when returning
Blind navigation to driver location
Navigation control signal
Feed back
Send goal
Send status
Navigati on node
Map
GMAPPING
Map
No
Do you navigate to the driver's point?
Yes
Turn on speech recognition
Voice signal control
Turn On
Voice node
data transmission
Does the robot receive a regression signal?
No
Yes
Yes
Blocking voice nodes
Navigati on node
Send goal
Feed back
Ready to go back to the observation point outside the door
Navigation control signal
No
Send status
Whether to reach the observation point outside the door
Perform GMAPPING2AMCL switching
Yes
AMCL
MAP
Navigati on node
Send goal
Feed back
Ready to return to the navigation initial servo pose
Navigation control signal
No
Send status
Has the robot reached the initial pose?
Yes
All tasks completed?
Yes
Stop servo

Figure 1 Task implementation process

From the above figure, we can see that we can complete the whole task mainly into the following stages:

1. First navigate to the initial point and wait for the voice to wake up (voice)
2. After the voice wakes up, it starts to recognize people (visual)
3. The location where you start to grab the object after identifying the person (navigation)

4. Start grabbing the object (mechanical arm) after reaching the location where the object is grabbed

5. Ready to navigate back to the original call point after the crawl is complete (navigation)

6. After reaching the call point, give clothes and an umbrella to wait for the voice to wake up and prepare to go out (voice & & mechanical arm | | manual assist)

7. After waking up, go out and reach the door identification point (navigation)

8. Start recognition at the door recognition point (visual)

9. After reaching the driver's location, arrive at the driver and turn on the voice recycling umbrella to wait for the return signal (speech && robotic arm || manual assist)

10. After receiving the return signal, prepare to return to the initial servo point for a new round of servo (navigation)

Robot system design framework

The picture above is the overall framework of our robot system design.

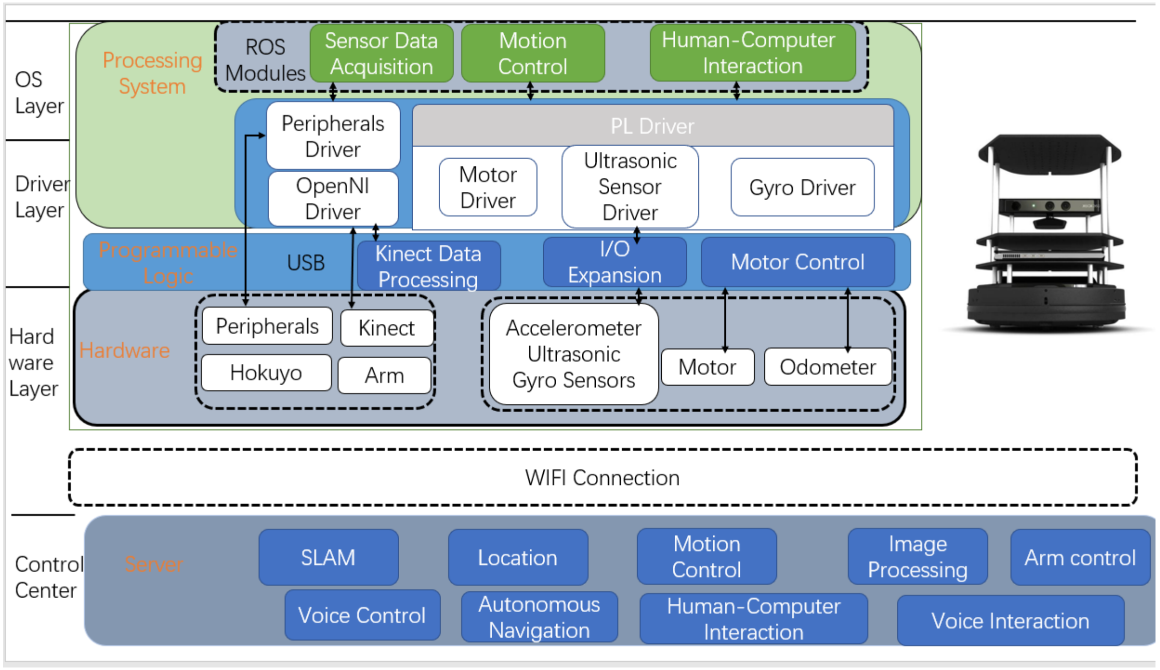


Figure 2 *Robot system design framework*

3 *Robot hardware platform*

distance sensor

In order to make the map more accurate and navigation more accurate, we chose to use Lidar navigation for navigation and mapping. Through the design we added the following laser radar driver, and finally completed the map-building related process with *gmapping* algorithm. Its main structure is as follows:

Figure 3 *distance sensor*

Vision sensor

Since the task requires us to identify people's information through visual algorithms, we use *astra* as our vision sensor.

Its main structure is shown below:

Figure 4 *Vision sensor*

Robot structure

The robot model we designed is as follows:

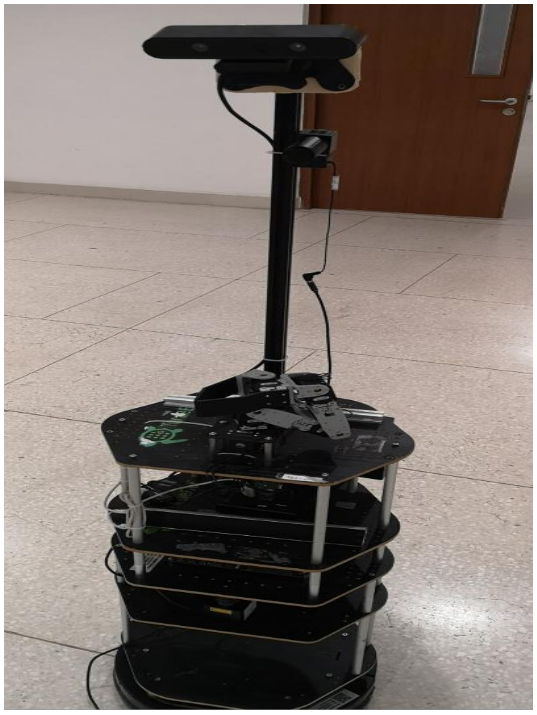


Figure 5 Robot hardware platform

4 *Robot Software Contributions*

Navigation and SLAM

SLAM Algorithm Optimization

When we initially tried to use the system's original *gmapping*
navigation, we found that the mapping effect was not good, and it had a great influence on the subsequent navigation. So we proposed navigation based on Kalman filtering algorithm. Below we give a comparison of the principle analysis and the actual operation results.

Principle Analysis of EKF_SLAM Algorithm

EKF Principle

The self-positioning process of mobile robots is a nonlinear process. From a statistical point of view, the SLAM problem is a filtering problem. EKF can remove the noise in the system well when the system noise and measurement noise obey the normal distribution. The interference also achieves the best estimate, so Smith and Cheeseman proposed the EKF-SLAM algorithm using EKF to estimate the SLAM problem. The block diagram of the algorithm is shown below.

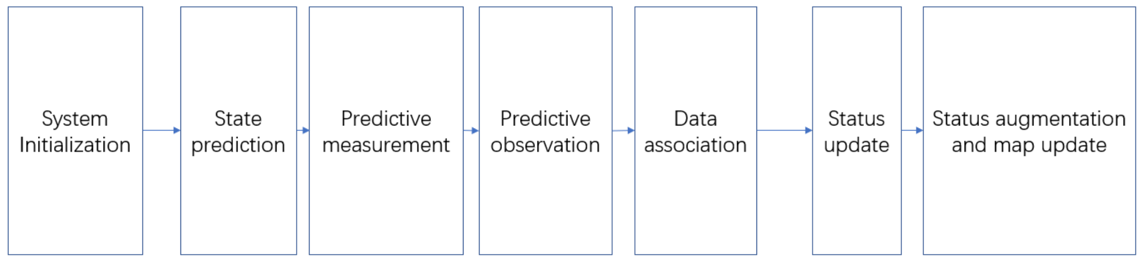


Figure 6 $EKF - SLAM\ Algorithm\ Process$

The extended Kalman filter algorithm is a commonly used nonlinear filtering method. The principle is to perform Taylor expansion on the system state equation and measurement equation at the prediction point $x_{k|k-1}$, and ignore all nonlinear expansion high order. The term thus achieves a linearized approximation of the nonlinear equation. The commonly used nonlinear discrete stochastic system models are as follows:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w(t) \\ z_k &= h(x_k) + v(t) \end{aligned} \qquad (1)$$

Where w, v represents the $0 - Mean\ Gaussian\ Noise$ added during the operation of the nonlinear system. Linearizing the system equation is expressed as:

$$\begin{aligned} x_k &\approx \hat{x}_k + J\left(x_{k-1} - \hat{x}_{k-1}\right) + W\omega_{k-1} \\ z_k &\approx \hat{z}_k + H\left(x_k - \hat{x}_k\right) + Vv_k \end{aligned} \qquad (2)$$

In the above formula $J = J(k) = \left.\frac{\partial f}{\partial x}\right|_{(x_k,u_k)}, H = H(k) = \left.\frac{\partial h}{\partial x}\right|_{x=x_k}, W = W(k) = \left.\frac{\partial f}{\partial \omega}\right|_{(x_k,u_k)}, V = V(k) = \left.\frac{\partial h}{\partial v}\right|_{x=x_k}, \hat{z}_k$

represents an observation, and $\hat{x}_k$ represents the predicted value. The initial time filter is set to:

$$\hat{x}_0 = E(x_0), P_0 = E\quad\left[x_0 - \hat{x}_0\right)\left(x_0 - \hat{x}_0\right)^{\mathrm{T}}] \qquad (3)$$

The prediction update can be performed by the first three sub-iterative operations below, and the status update can be performed by the following four sub-iterative operations.

$$\begin{aligned}
&x_{k+1|k} = f\left(x_{k|k}, u_k\right)\\
&P_{k+1|k} = JP_{k|k}J^{\mathrm{T}} + Q_k\\
&S_{k+1} = H_{k+1}P_{k+1|k}H_{k+1}^{\mathrm{T}} + R_{k+1}\\
&W_{k+1} = P_{k+1|k}H_{k+1}^{\mathrm{T}}S_{k+1}^{-1}\\
&\vartheta_{k+1} = z_{k+1} - h\left(x_{k+1|k}\right)\\
&x_{k+1|k+1} = P_{k+1|k} - W_{k+1}S_{k+1}\\
&P_{k+1|k+1} = P_{k+1|k} - W_{k+1}S_{k+1}W_{k+1}^{\mathrm{T}}
\end{aligned} \qquad (4)$$

$\boldsymbol{EKF - SLAM\ Algorithm}$

The EKF-based SLAM algorithm includes two processes: motion update and observation update, which are descriptions of the system's motion model and observation model, respectively. The motion update process uses each sensor data as an input control amount to predict and estimate the current position of the robot. The observation update is a process in which the mobile robot observes the feature points in the current environment and matches the partial map while updating the position of the robot in the map. Usually the robot SLAM stores the pose information x_v, and the map feature $m_i(x_i, y_i)$ in a separate state vector x_{ml}:

$$x_{ml}(k) = \begin{bmatrix} x_v(k) \\ m^i(k) \end{bmatrix} \qquad (5)$$

Where $x_v(k)$ represents the pose of the mobile robot at time k, and $m_n^i(k)$ represents the set of n map point features at time k. The covariance matrix between the features and poses of the system is:

$$P_{ml}(k) = \begin{bmatrix} p_{tv}(k) & p_{rm}(k) \\ p_{mv}(k) & p_{mm}(k) \end{bmatrix} \qquad (6)$$

The entire map information is $M^B = (x_{ml}, P_{ml})$

In the prediction of the state of the robot by the EKF algorithm (4), the prediction of the feature position of the map is included: $\hat{z}$, and the mobile robot performs data association matching through the observation value of the external sensor z, and the mobile robot passes the $The matching information of z$ and $\hat{z}$ is self-positioned and the map is updated.

Finally, we combined the *gmapping* algorithm to achieve the complete construction of the map.

Algorithm Utility Analysis

Below is a map that we built directly with the *gmapping* algorithm without adding *EKF Filtering*.

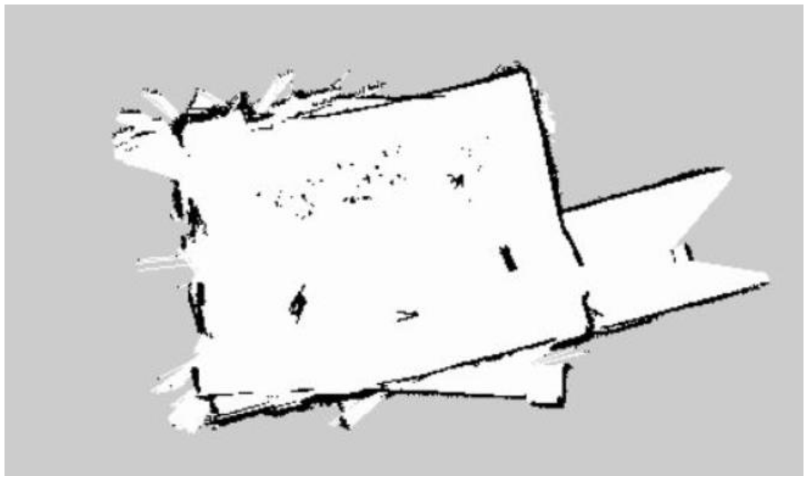


Figure 7 Map created without EKF filtering

It can be seen that the mapping effect is poor. The following is the effect of the mapping in the same environment that we implemented with*EKF Filtering*:

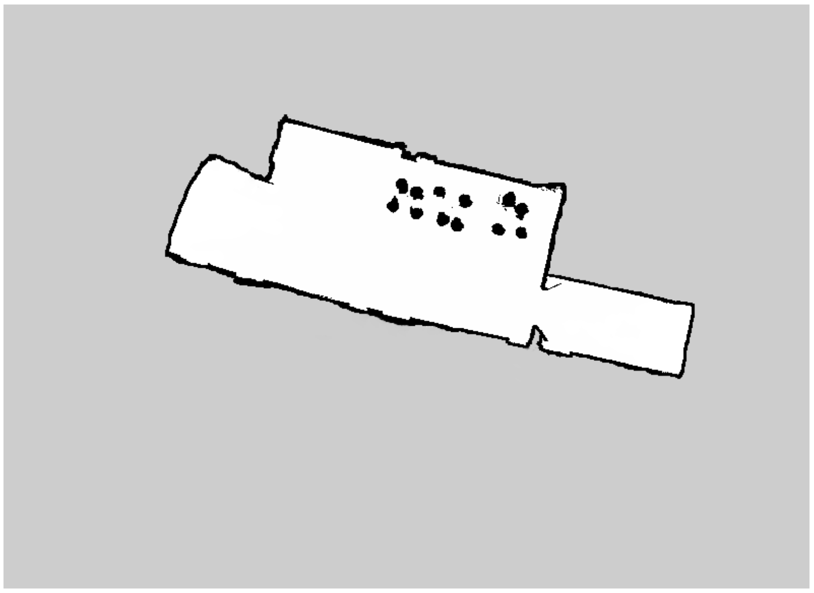


Figure 8Map created with EKF filtering

From the above figure, we can see that the SLAM construction effect after adding the EKF algorithm has been significantly improved, and the finally established image can be closed.

Navigation

Automated Real – Time Pose Output

In our design process, the pose latching technique is used many times, that is, the pose of a certain moment is saved to restore the original pose state in the same global coordinate system in the future, thereby achieving full state restoration.

The main purpose is to use the conversion data of the map coordinate system and the base_link coordinate system, and transmit the corresponding coordinate system to transform the data, thereby obtaining the real-time pose information of the robot at any time.

Automated Positioner

Based on the pose and latching, we hope that $rviz's$ pose positioning function can be realized without human operation, so that we can automatically perform $rviz$ pose correction in the case of determining the pose of the robot.

The message diagram of its is as follows:

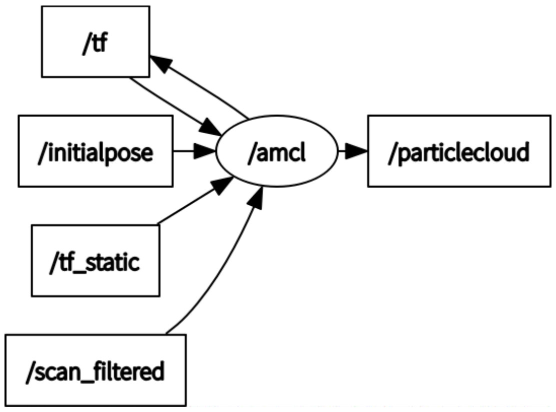


Figure 9AMCL node message flow diagram

Mainly by releasing the topic information named `"initialpose"`, the pose state of the robot is restored by the originally locked pose information, and finally the success rate of the $AMCL$ navigation and positioning can be improved.

Blind Navigation Technology

The initial *gmapping* combined with the *move_base* navigation cannot reach the unknown area, that is, the area without the map cannot navigate autonomously.

So by modifying the parameter interface of *move_base* and combining the coordinate input of visual recognition, we finally realized the navigation function in the case where the map is unknown.

Map Navigation and No − Map Navigation Switching Technology

Move_base Navigation Frame

The main framework of *move_base* is as follows:

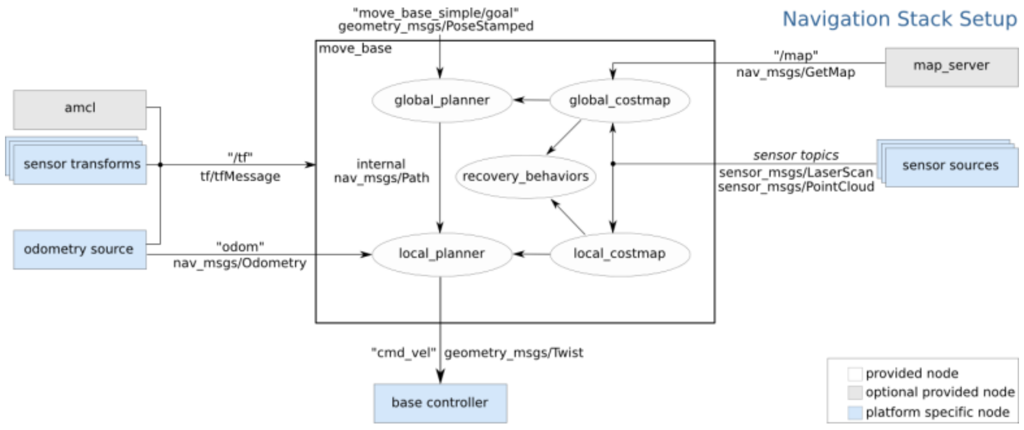


Figure 10 *move_base* Basic Framework

In the navigation of *ros*, we generally use the keyboard control to combine *gmapping* for *SLAM* and build the map, use *map_server* to store the map we just created, and then open *AMCL* with the map that has been built. Further navigation. We hope to update the map in real time while navigating, and keep the original map intact, which is convenient for secondary use. However, if we open *gmapping* directly, the updated map will empty the original map, which will not be able to maintain the original map. So we designed a map navigation using *AMCL* and *map_server* and a no-map navigation using *gmapping* and *move_base*, and built a bridge between the two states to achieve map navigation. Smooth switching with and without map navigation can finally achieve the effect of the secondary servo required by *Farewell Task*.

Map Navigation Node Message Map

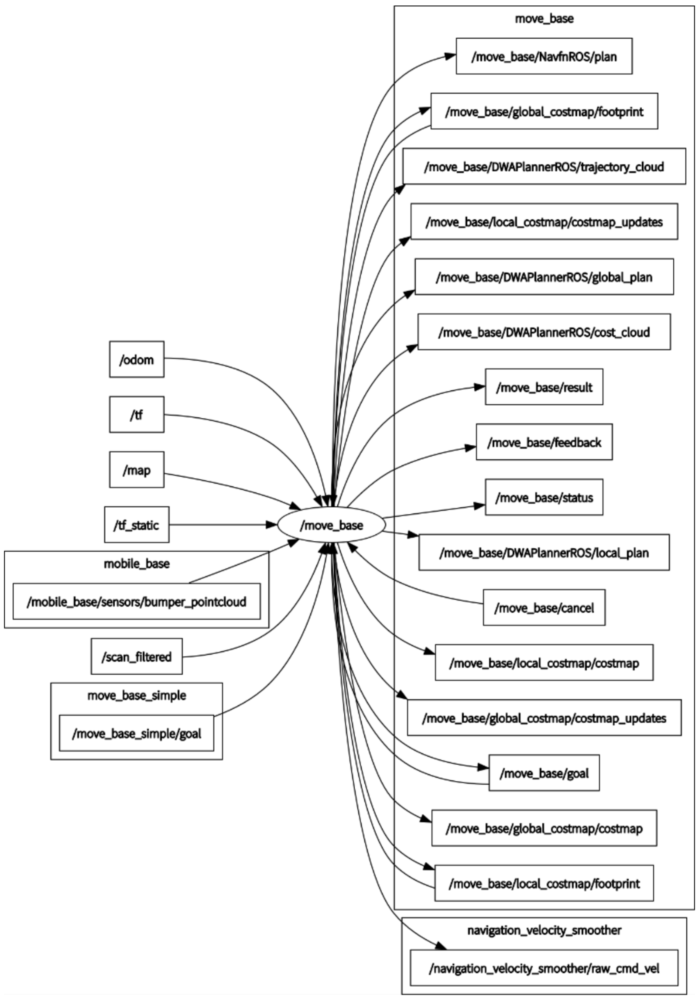

move_base
/move_base/NavfnROS/plan
/move_base/global_costmap/footprint
/move_base/DWAPlannerROS/trajectory_cloud
/move_base/local_costmap/costmap_updates
/move_base/DWAPlannerROS/global_plan
/move_base/DWAPlannerROS/cost_cloud
/odom
/move_base/result
/tf
/move_base/feedback
/map
/move_base/status
/tf_static
/move_base
mobile_base
/move_base/DWAPlannerROS/local_plan
/mobile_base/sensors/bumper_pointcloud
/move_base/cancel
/scan_filtered
/move_base/local_costmap/costmap
move_base_simple
/move_base/global_costmap/costmap_updates
/move_base_simple/goal
/move_base/goal
/move_base/global_costmap/costmap
/move_base/local_costmap/footprint
navigation_velocity_smoother
/navigation_velocity_smoother/raw_cmd_vel

Figure 11 *move_base* message frame diagram

From the above figure, we can see that we use the filtered laser information and provide a static map to design the entire contents of the map navigation.

Dynamic State Switching Technology

In order to achieve the following process:

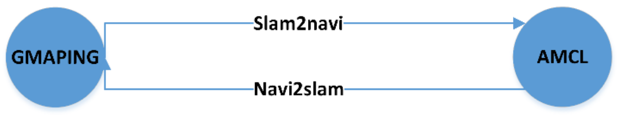


Figure 12 Navigation state transition diagram

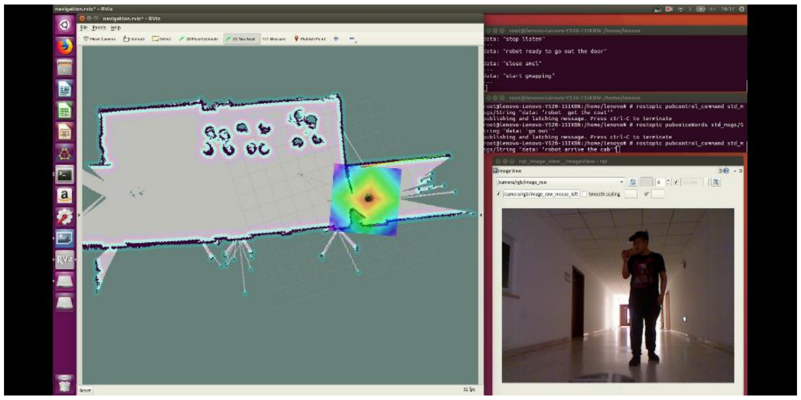
To complete the state switching, we have designed the dynamic state switching algorithm as follows:

First, we design an empty node to easily close the corresponding state and turn on another state.

So we can complete the closing of the *gmapping* state.

Then by interacting directly with the system, dynamic state switching is achieved, so we can achieve complete unattended operation.

State of The RVIZ Before Switching



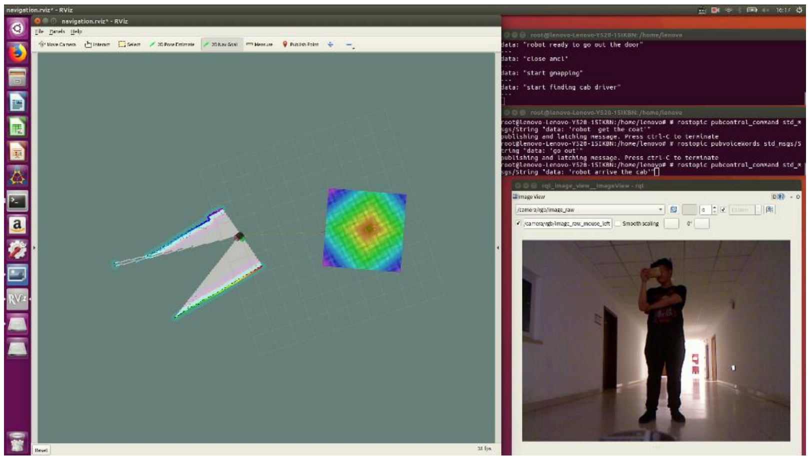
```
data: "stop listen"
---
data: "robot ready to go out the door"
---
data: "close amcl"
---
data: "start gmapping"
---
```

```
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubcontrol_command std_m
sgs/String "data: 'robot  get the coat'"
publishing and latching message. Press ctrl-C to terminate
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubvoiceWords std_msgs/S
tring "data: 'go out'"
publishing and latching message. Press ctrl-C to terminate
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubcontrol_command std_m
sgs/String "data: 'robot arrive the cab'"
```

Figure 13 Operation diagram before state transition

As can be seen from the above figure, when *AMCL* is turned on, we are navigating based on the map.

RVIZ State After Switching



```
data: "robot ready to go out the door"
---
data: "close amcl"
---
data: "start gmapping"
---
data: "start finding cab driver"
---
```

```
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubcontrol_command std_m
sgs/String "data: 'robot  get the coat'"
publishing and latching message. Press ctrl-C to terminate
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubvoiceWords std_msgs/S
tring "data: 'go out'"
publishing and latching message. Press ctrl-C to terminate
root@lenovo-Lenovo-Y520-15IKBN:/home/lenovo# # rostopic pubcontrol_command std_m
sgs/String "data: 'robot arrive the cab'"
```

Figure 14 Operational diagram after state transition

After turning on *gmapping*, we implemented blind navigation for building maps in real time.

voice

Keda Xunfei Speech Recognition API Optimization

We mainly use the speech recognition system of *Keda Xunfei*. Among them, we complete the voice control through a series of processes such as voice reception, string filtering, command recognition and voice blocking.

Based on the *Keda Xunfei* API, we designed the ROS interface to design the corresponding speech recognition and speech recognition wake-up operations. The main node relationship diagram is as follows:

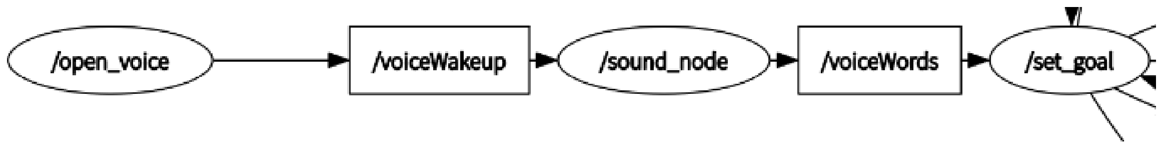


Figure 15 Voice node control flow graph

And`/sound_node` can send the identified information to our core node.

Voice Blocking

In order to reduce the computational pressure of the computer during voice control, improve the real-time performance of the robot system and avoid the interference of the wrong voice control and noise on the control system, we realized the automatic opening and closing of the voice recognition by adapting the *Keda Xunfei* voice API.

From the figure15, we can see that we turn on and off speech recognition through the `/open_voice` node control`/sound_node`(the modified node of the *Keda Xunfei* voice API).

Speech Filtering

Because the string format recognized by *Keda Xunfei* is not standardized, it is difficult to process, and Chinese characters are often recognized, which tends to cause the node to close. Therefore, we have added a string filtering algorithm, mainly by designing keyword recognition technology and removing noise. To improve the accuracy of our speech recognition.

Through the relevant tests, our algorithm can recognize the corresponding control signals regardless of the voice recognition input of noise, noise and Chinese characters, so as to accurately initiate the control command.

5 *Technology and Scientific Contributions*

Finite State Machine Control

We mainly design a set of finite state machines that can be self-starting. The flow of the entire finite state machine is as follows:

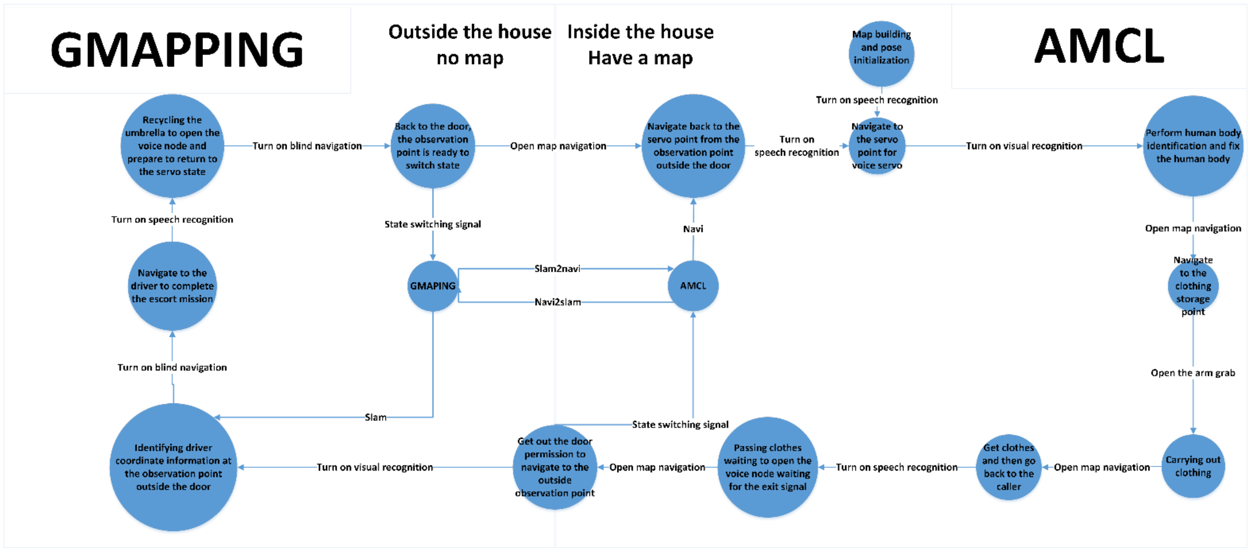

GMAPPING
Outside the house no map
Inside the house Have a map
AMCL
Map building and pose initialization
Turn on speech recognition
Recycling the umbrella to open the voice node and prepare to return to the servo state
Turn on blind navigation
Back to the door, the observation point is ready to switch state
Open map navigation
Navigate back to the servo point from the observation point outside the door
Turn on speech recognition
Navigate to the servo point for voice servo
Turn on visual recognition
Perform human body identification and fix the human body
Turn on speech recognition
State switching signal
Navi
Open map navigation
Navigate to the driver to complete the escort mission
GMAPING
Slam2navi
Navi2slam
AMCL
Navigate to the clothing storage point
Turn on blind navigation
Open the arm grab
Slam
State switching signal
Identifying driver coordinate information at the observation point outside the door
Turn on visual recognition
Get out the door permission to navigate to the outside observation point
Open map navigation
Passing clothes waiting to open the voice node waiting for the exit signal
Turn on speech recognition
Get clothes and then go back to the caller
Open map navigation
Carrying out clothing

Figure 16Loopback finite state machine design diagram

From the above figure we can see that we have adopted loopback control. At the beginning and end of the mission, the state of the robot is guaranteed to be consistent, including the consistent pose and the state of the algorithm. That is, before each task is executed, the robot is in the same state (that is, the state of the map navigation and voice servo is turned on).

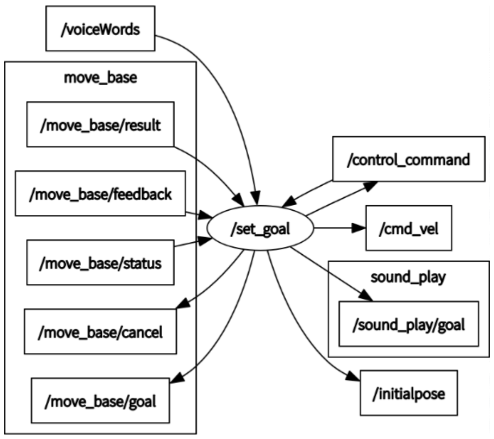


Figure 17Core node message flow diagram

As can be seen from the above figure, we mainly use the voice signal control combined with the state signal to control the two-way control to ensure that the voice node can be blocked at the moment when the voice is not needed, and the robustness of the control system is improved.

6 *Summary and Outlook*

Summary

We mainly complete the whole task of farewell by designing a complete closed-loop control system, which proposes multi-sensor fusion, blind navigation combined with deep learning visual positioning, and combined with $Keda\ Xunfei$ intelligent voice controller and voice response. For robot software development, we first adopted a dual-controlled algorithm execution framework and implemented a navigation-centric multi-interface multitasking execution system.

Outlook

We hope to further optimize the path planning for blind navigation by combining the RRT algorithm.

7 *References*

[1]Xu Yulong. Research on ROS-based mobile robot construction and navigation technology [D]. Harbin Institute of Technology, 2017.

[2]Zhu Jingquan. Research on interactive map construction and path planning technology for service robots [D]. Harbin Institute of Technology, 2017.

[3]Gu Qiyang. Design and implementation of ROS-based home service robot SLAM [D]. Northwest Normal University, 2016.

[4] Tian Liangliang. ROS-based exploration and environment modeling of unknown regions of mobile robots [D]. Northeastern University, 2015.

[5] Hu Chunxu, Xiong Wei, Ren Wei, He Dingxin. Positioning and Navigation of Indoor Mobile Robot Based on Embedded System[J]. Journal of Huazhong University of Science and Technology(Natural Science Edition), 2013, 41(S1): 254-257 +266.