SOUTHEAST UNIVERSITY

Human-Robot-Interaction

Final Project

Submitted by

Yongqiang Zhao
ID: 08117102
Robotics Engineering

Suprevised by

Prof. Kun Qian

April, 2020

1 Overall completion of the group

Our team completed simulation, objects identification, face recognition, voice generation and speech recognition, etc., and concentrated on the main process.
In a word, we fully realized the experiment of human/objects recognition and voice interaction.

Name	ID	actually completed work, proportion of tasks undertaken
Yongqiang Zhao	08117102	simulation, 25%
Zhihai Bi	08117106	face recognition, 25%
Shaopu Song	08117122	voice interaction, 25%
Yu Du	08117125	objects identification & main framework, 25%

Table 1: team members and division of labor

2 Overall principle and design

- **Main Framework**

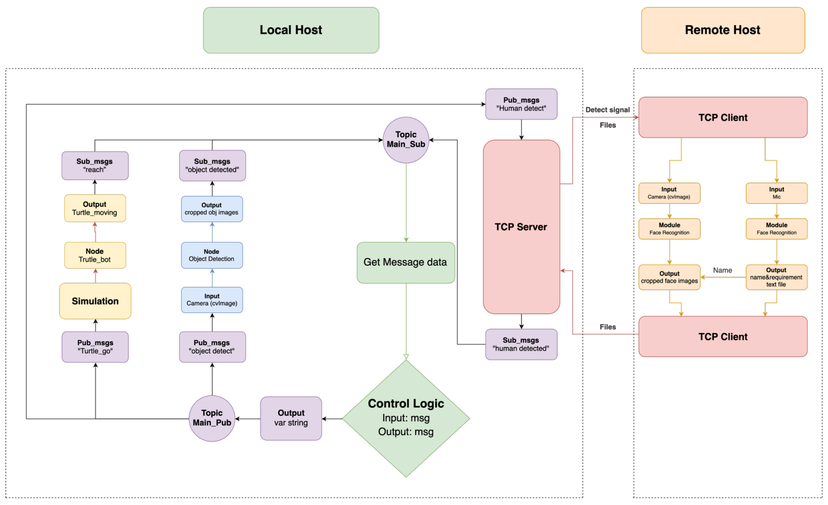


Figure 1: main framework

And our messages are defined as follows

$Main\_Sub\ (Send\ by)$	$\Rightarrow$	$Main\_Pub\ (Send\ to)$
$'Init'$	$\rightarrow$	$Turtle\_start\ (SimNode)$
$Turtle\_reach\_human1\ (SimNode)$	$\rightarrow$	$Human\_detect\ (TcpNode)$
$Human1\ (TcpNode)$	$\rightarrow$	$Human\_find2\ (SimNode)$
$Human2\ (TcpNode)$	$\rightarrow$	$Human\_find3\ (SimNode)$
$Human3\ (TcpNode)$	$\rightarrow$	$Turtle\_go\ (SimNode)$
$Turtle\_reach\_goods\ (SimNode)$	$\rightarrow$	$Object\_detect\ (ObjNode)$
$Object\_detected\ (ObjNode)$	$\rightarrow$	$Turtle\_back\ (SimNode)$
$Turtle\_reach\_human2\ (SimNode)$	$\rightarrow$	$Send\_object\ (TcpNode)$

Figure 2: routing table

- **Simulation**
  We used the gazebo simulator to build a simulated Turtlebot robot under ROS, and then built a map through Gmapping and presented it with Rviz. Finally, we completed the task through multi-point navigation.

Figure 3: gazebo

Figure 4: Turtlebot

- **Objects Identification**
  We used yolov3 combining the list of ten kinds of objects and the weight of the trained yolov3 which were provided by the teacher. And finally, we realized real time detection.

- **Face Recognition**
  After comparing Face_Recognition and Baidu API, we decided to use Baidu API, and the flow chart is as follows

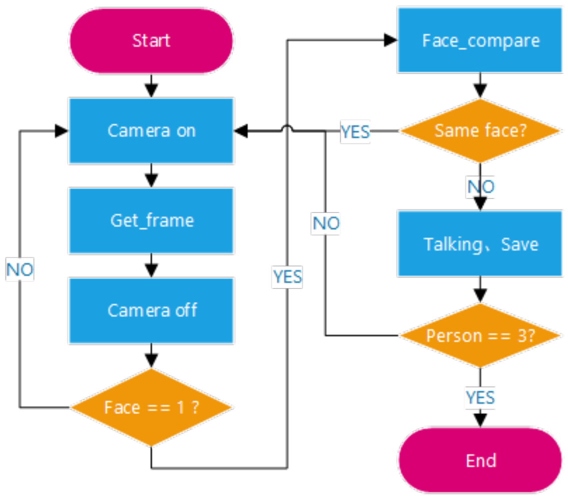


Figure 5: the flow chart of face recognition

- **Voice Interaction**
  By comparing Baidu API and Google API, we finally adopted Google API and added natural language processing.
  The structure diagram is drawn as follows

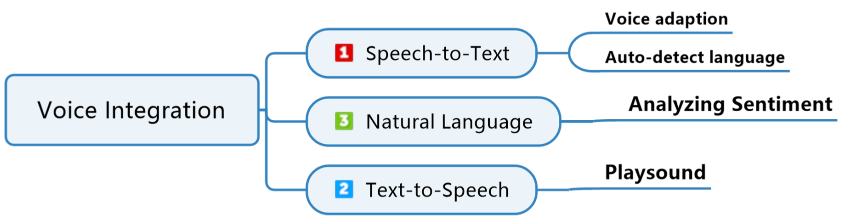


Figure 6: the structure diagram of voice interaction

And the logic diagram is drawn as follows

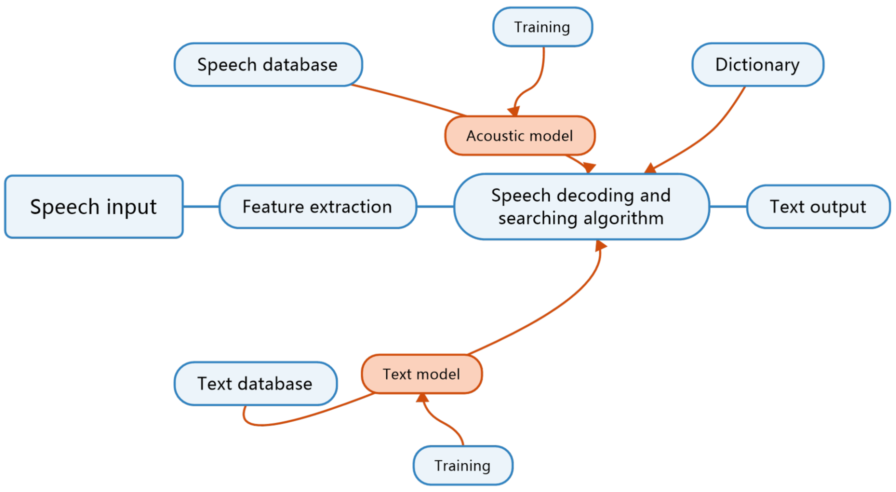


Figure 7: the logic diagram of voice interaction

- **Socket Thread**

  We placed simulation, objects recognition and main process on one computer, and placed face recognition and voice interaction on another computer. The two computers passed information through TCP/IP protocol.

  Prerequisite: WAN IP Configuration: DMZ Host & NAT(Port Mapping)

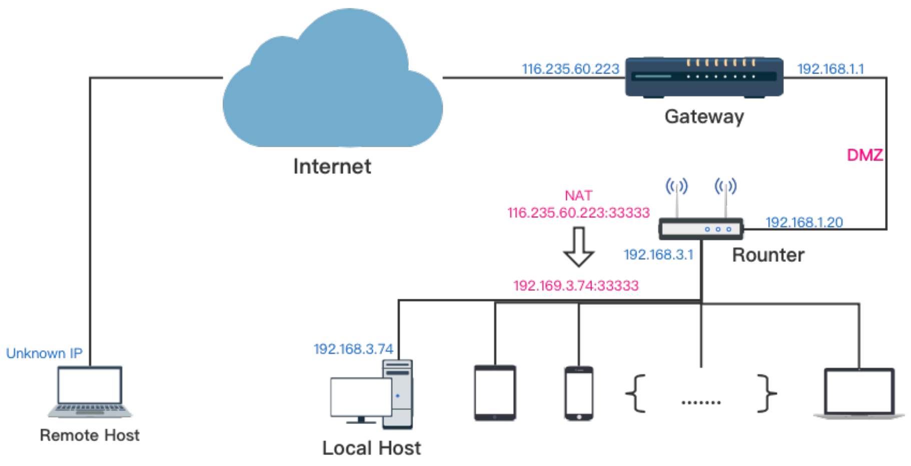


Figure 8: socket thread

3 Specific development and implementation

3.1 My Part

3.1.1 Build a simulation environment

- **Edit a room model**

  Use the building editor function of gazebo

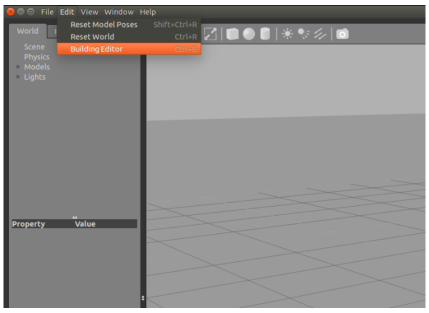


Figure 9: building editor

Use the four control buttons of add wall, add windows, add door and add star in the upper left corner to build the environment

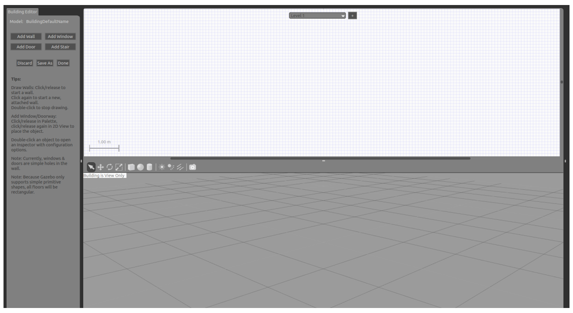
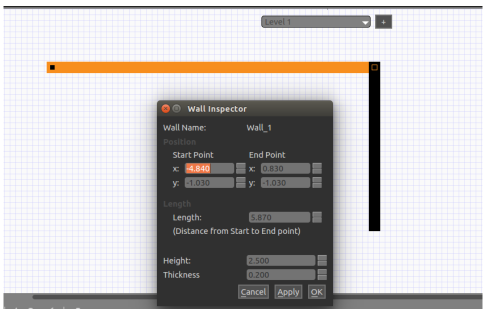


Figure 10: four control buttons

Right click on the wall to open the wall inspector editor, which is used to edit the length of the wall and other information



Wall Inspector

Wall Name: Wall_1

Position

	Start Point	End Point
x:	-4.840	0.830
y:	-1.030	-1.030

Length

Length: 5.870

(Distance from Start to End point)

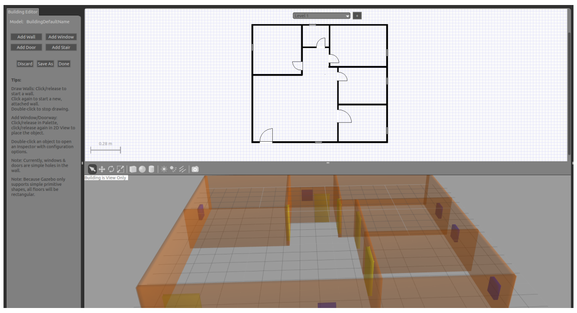
Height: 2.500

Thickness 0.200

Cancel Apply OK

Figure 11: wall inspector editor

Finally, a room model has been built, like the following



Model: BuildingDefaultName

Tips:

Draw Walls: Click/release to start a wall.
Click again to start a new, attached wall.
Double-click to stop drawing.

Add Window/Doorway: Click/release in Palette, click/release again in 2D View to place the object.

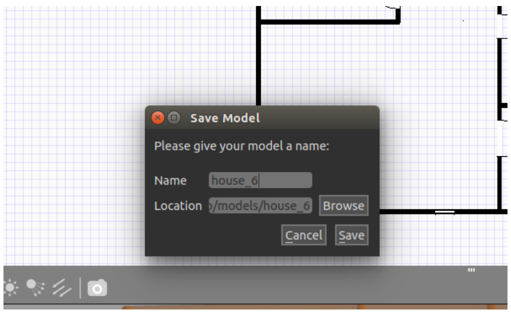
Double-click an object to open an Inspector with configuration options.

Note: Currently, windows & doors are simple holes in the wall.

Note: Because Gazebo only supports simple primitive shapes, all floors will be rectangular.

Figure 12: room model

Before saving, first open the hidden directory with Ctrl+H under the home directory, and create a new folder under the path of .gazebo/models to save the built model. Note that the model name should be the same as the file name. Save as follows,



Please give your model a name:

Name house_6

Location /models/house_6

Figure 13: save model

- **Edit a world file**
  Open the gazebo editor and select the model just built in the insert tab. By using the existing model of gazebo, we can continue to improve our environment and build a world of our own.
  Finally, we used the models which are the room model, sun, cabinet, bookshelf person_standing and ISCAS_groundplane(from ROS learning package of Chinese Academy of Sciences).
  The world file is edited as follows

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
  <scene>
    <ambient>0.68 0.68 0.68 1.0</ambient>
    <sky>
      <sunrise/>
      <clouds>
        <speed>12</speed>
    </clouds>
    </sky>
  </scene>
    <physics type='ode'>
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1</real_time_factor>
      <real_time_update_rate>1000</real_time_update_rate>
      <gravity>0 0 -9.8</gravity>
    </physics>

    <include>
      <uri>model://ISCAS_groundplane</uri>
    </include>
    <include>
      <uri>model://sun</uri>
    </include>
    <include>
      <uri>model://our_house</uri>
      <pose>0 -1.0 0 0 0 0</pose>
    </include>
    <include>
      <uri>model://cabinet</uri>
      <pose>-2.32 -2.965 0 0 0 0</pose>
    </include>
    <include>
      <uri>model://bookshelf</uri>
      <pose>3.3496 -4.399 0 0 0 0</pose>
    </include>
    <include>
      <uri>model://person_standing</uri>
      <name>person_standing</name>
      <pose>2.238798 -3.10881 0 0 0 -2.355</pose>
    </include>
    <include>
      <uri>model://person_standing</uri>
      <name>person_standing_0</name>
      <pose>0.471 -3.492 0 0 0 2.70</pose>
    </include>
    <include>
      <uri>model://person_standing</uri>
      <name>person_standing_1</name>
      <pose>2.746 -1.652 0 0 0 -1.47</pose>
    </include>
  </world>
</sdf>
```

- **Edit a launch file**
  Use the launch file to directly call the world we just set up. And add the configuration related to Turtlebot
  The final world is as follows

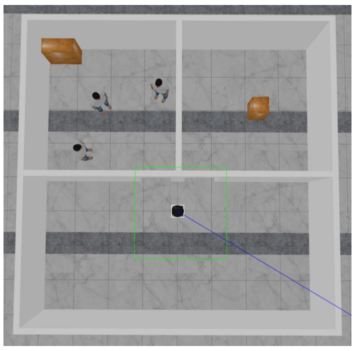


Figure 14: final world

3.1.2 Simulation package configuration

Install the related dependencies of Turtlebot.

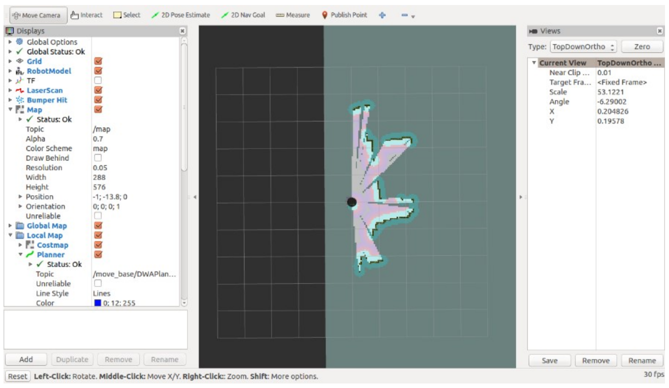
```
sudo apt-get update sudo apt-get install ros-kinetic-turtlebot
    ros-kinetic-turtlebot-apps ros-kinetic-turtlebot-interactions
    ros-kinetic-kobuki-ftdi ros-kinetic-ar-track-alvar-msgs
    ros-kinetic-turtlebot-simulator
```

Put related packages in ROS workspace, then compile it.

3.1.3 Mapping

Use the official gmapping package of ROS on Turtlebot, and the laser ranging sensor of Turtlebot is used for simultaneous location and mapping (SLAM) in a strange environment. Here is the process

- Run roslaunch robot_sim_demo turtlebot_world.launch to open the simulation environment;
- Start keyboard control program, run roslaunch turtlebot_teleop keyboard_teleop.launch;
- Building maps with gmapping, run roslaunch turtlebot_navigation gmapping_demo.launch;
- Run roslaunch turtlebot_rviz_launchers view_navigation.launch to open the rviz interface, as shown in the figure below



```
Global Options
Global Status: Ok
Grid                ✓
RobotModel          ✓
TF                  ☐
LaserScan           ✓
Bumper Hit          ✓
Map                 ✓
  Status: Ok
  Topic             /map
  Alpha             0.7
  Color Scheme      map
  Draw Behind       ☐
  Resolution        0.05
  Width             288
  Height            576
  Position          -1; -13.8; 0
  Orientation       0; 0; 0; 1
  Unreliable        ☐
Global Map          ✓
Local Map           ✓
  Costmap           ✓
  Planner           ✓
    Status: Ok
    Topic           /move_base/DWAPlan...
    Unreliable      ☐
    Line Style      Lines
    Color           0; 12; 255
```

```
Type: TopDownOrtho     Zero
Current View    TopDownOrtho ...
  Near Clip ...   0.01
  Target Fra...   <Fixed Frame>
  Scale           53.1221
  Angle           -6.29002
  X               0.204826
  Y               0.19578
```

Figure 15: turtlebot_rviz

The turnlebot starts from the origin of map coordinate system, and its positive direction is the positive direction of X axis. Click the keyboard control terminal opened in step 2, and control the robot movement through the keyboard to complete the map construction of the whole environment;

- After building the map, run rosrun map_server map_saver -f /filepath in the terminal, and filepath is the save path.

3.1.4 Positioning and navigation

According to the established map, use global positioning(amcl) and navigation(move_base) of the laser ranging sensor.

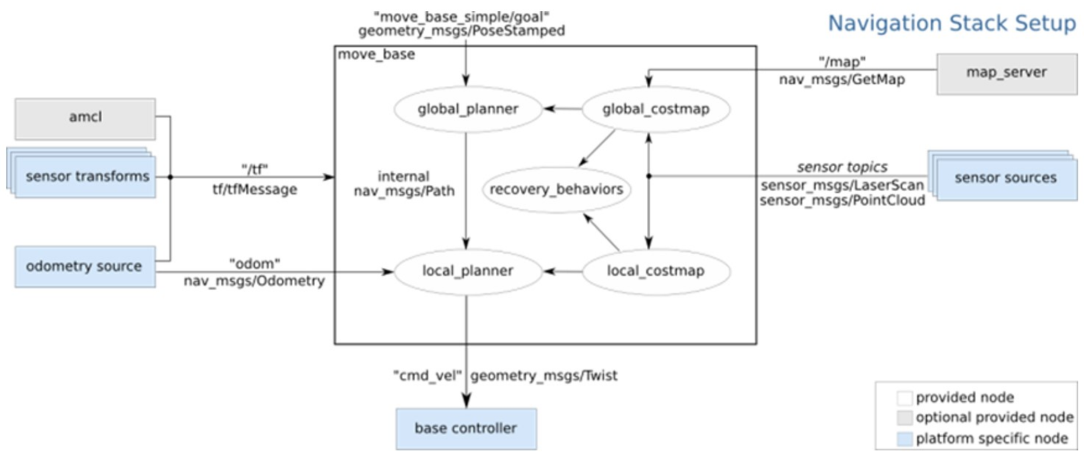


Figure 16: organization of navigation function package

As shown in the figure above, In the white box are the components that ROS has prepared to use(move_base), in the gray box are the optional components of ROS(amcl and map_server), and in the blue is the components on the robot platform that users need to provide(sensor transforms, odometry source, sensor source, base controller).
"move_base" is the central hub of robot path planning under ROS. It subscribes to the data of laser sensor, map, amcl and so on, then plans the global and local path, then transforms the path into the speed information of the robot, and finally realizes the robot navigation. Here, we mainly talk about the core part of move_base, that is the content in the middle box.
First of all, we need to understand global cost map and local cost map. In order to understand the concept of cost map, we need to know which map that the path is generated by the planning algorithm when the robot is planning the path. It relies on a global map of the environment built by gmapping scanning, but it is not possible to rely on only one original global map. Because this map is static, we can't update the obstacle information on the map at any time. In the real environment, there will always be unexpected new obstacles in the current map, or old obstacles have been removed from the environment map, so we need to update this map at any time. At the same time, since the default map is a black white gray three-color map, only the obstacle area, free moving area and unexplored area will be marked. Robot path planning in such a map will lead to unsafe path planning, because our robot needs to maintain a certain safe buffer distance with obstacles when moving, so that the robot is safer when moving in the current map. As shown in figure, it is an original global map built through gmapping. Among them, the white area represents the area that can move freely, the black area represents the obstacle area, and the gray area represents the unexplored area, that is, the unknown area. Costmap is simply to carry out various processing on this map, which is convenient for us to carry out path planning later. In ROS, costmap_2D is used to implement costmap. This package implements two new maps on the original map. One is local cost map, the other is global cost map. Two cost maps are prepared for local path planning and the other for global path planning.
After understanding the global cost map and the local cost map, the global planner and the local planner, as the name implies, generate the global path and the local path based on the

global costmap and the local costmap respectively. The local path generated by the local planner will fit the global path as much as possible and take into account the obstacles that may appear at any time. The common global planning algorithms in ROS include Djikstra and A* algorithm, while the local planning algorithms include trajectory rollout and DWA algorithm. Finally, recovery_behaviors will take some recovery actions when the robot is stuck in obstacles and other special circumstances.

3.1.5 Adjust posture

Set the linear speed to 0 and the angular speed to 5 deg/s in the turnlebot simulation environment in the callback function. When receiving the pose adjustment data sent by the main process, the robot starts to rotate in place to adjust the angle.
The key codes are as follows

```
def adjustcallback(data):
cmd_vel = rospy.Publisher('cmd_vel_mux/input/navi', Twist, queue_size=10)
# 5 HZ
r = rospy.Rate(5)

angle = 5 * int(data.data / 5)
turn_cmd = Twist()
turn_cmd.linear.x = 0
turn_cmd.angular.z = radians(5);  # turn at 5 deg/s

rospy.loginfo("Turning")
for x in range(0, angle):
    cmd_vel.publish(turn_cmd)
    r.sleep()
return
```

3.1.6 Build a simulated person

At first, use the person model of gazebo, and then consider the person model which uses 3d max, blender or maya to build.

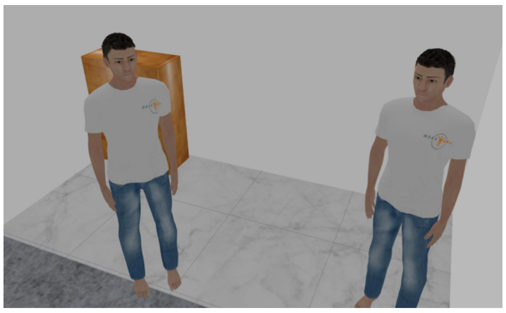


Figure 17: person model

3.1.7 Dock the main process

This module is completed earlier, so I use the method of simulating the main process to check the function first, and then it docks the ture main process well after modifying some node names.

3.2 The Others

- **Zhihai Bi** Used Baidu API to finish face recognition, met and solved many problems, and completed the task very well.
- **Shaopu Song** Used Google API combined with natural language processing to finish voice interaction, completed TCP/IP transmission and final verification with Yu Du, and completed the task very well.
- **Yu Du** Used yolov3 to realize real time detection, and write the main framework, completed TCP/IP transmission and final verification with Shaopu Song. And finally, completed the task very well.

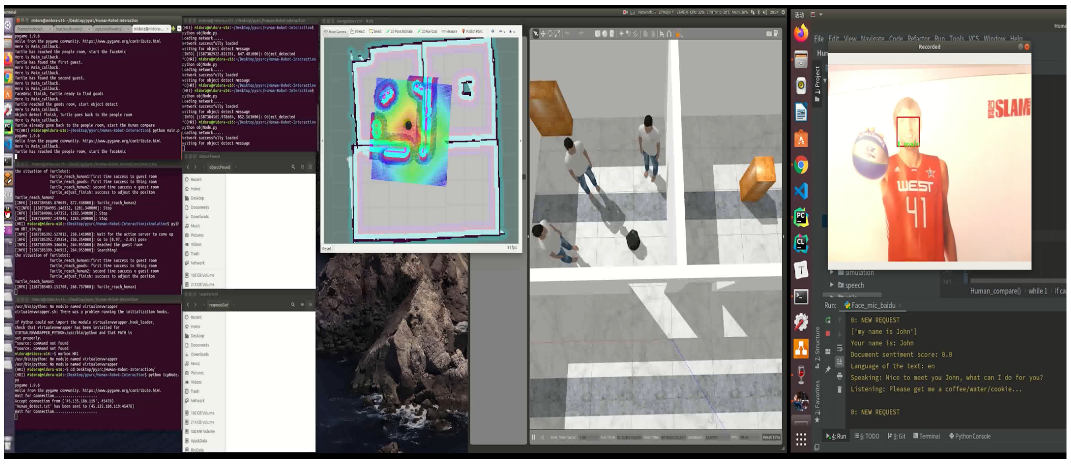
4 Experimental evaluation

4.1 Individual module testing

For the simulation control part of the turtlebot, open a new terminal in the HRI_sim.py file, enter the directory where the HRI_sim.py file is located, and input python HRI_sim.py. At this time, the turtlebot in the simulation environment will wait for the command signals of other parts (such as the main process, face recognition, voice interaction part, object recognition part, etc.);
Use talk.py and adjust.py to simulate other parts of the transmission instructions to test the function of this part, in which talk.py is used to simulate the main process, which is used to send the signal to specify the room position to the turtlebot, and adjust.py is used to send the turtlebot position adjustment signal, and open two new terminals to input python talk.py and python adjust.py respectively.
Enter 0 in the terminal where talk.py is located, that is, give the first instruction to the turtlebot to go to the room with three guests. At this time, in HRI_sim.py terminal, you can see that the turtlebot performs the action of going to the specific point of the room with guests. In the simulation environment, you can also see that the turtlebot is moving. After the turtlebot first arrives at the designated point of the room with guests, it will send 4 and then wait for the next instruction. If you input the angle information that should be adjusted in the terminal where adjust.py is located, (at present, the turnbot only realizes counterclockwise rotation, so the angle information given is 0°-360°, 0° is the front side of the turtlebot when the instruction is issued), At the terminal where HRI_sim.py is located, you can see the information that the turtlebot is adjusting. After the adjustment is completed, it will send out 7 signals. After three times of adjustment, input 1 at the talk.py terminal, and the turtlebot performs the action of going to another room to pick up the goods. At this time, you can see the relevant information at the HRI_sim.py terminal. After picking up the goods, input 2 at the talk.py terminal The turtlebot will take the guest's room and deliver the goods. After three times of the same posture adjustment, it finally completes the task.

4.2 Overall module testing

We have integrated face recognition and voice interaction, and configured and compiled it on a computer. At the same time, we also configure simulation, object recognition and main process to another computer, and then the two computers transmit information through TCP/IP protocol. After testing, each module has realized its own functions, and achieved the requirements of the experiment.



```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Here is Main_callback.
Turtle has reached the people room, start the face&mic
Here is Main_callback.
Turtle has found the first guest.
Here is Main_callback.
Here is Main_callback.
Turtle has found the second guest.
Here is Main_callback.
Here is Main_callback.
Face&mic finish, Turtle ready to find goods
Here is Main_callback.
Turtle reached the goods room, start object detect
Here is Main_callback.
Object detect finish, Turtle goes back to the people room
Here is Main_callback.
Turtle already gone back to the people room, start the Human compare
^C(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction$ python main.p
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Here is Main_callback.
Turtle has reached the people room, start the face&mic
```

```
the situation of Turtlebot:
                Turtle_reach_human1:first time success to guest room
                Turtle_reach_goods: first time success to thing room
                Turtle_reach_human2: second time success o guest room
                Turtle_adjust_finish: success to adjust the positon
Turtle_reach_human2
[INFO] [1587384585.870649, 872.438000]: Turtle_reach_human2
^C[INFO] [1587384995.146332, 1281.340000]: Stop
[INFO] [1587384996.147353, 1282.340000]: Stop
[INFO] [1587384997.147846, 1283.340000]: Stop
(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction/simulation$ pyth
on HRI_sim.py
[INFO] [1587385392.527012, 258.141000]: Wait for the action server to come up
[INFO] [1587385392.739354, 258.354000]: Go to (0.97, -2.05) pose
[INFO] [1587385399.346636, 264.955000]: Reached the guest room
[INFO] [1587385399.346953, 264.955000]: Searching!
the situation of Turtlebot:
                Turtle_reach_human1:first time success to guest room
                Turtle_reach_goods: first time success to thing room
                Turtle_reach_human2: second time success o guest room
                Turtle_adjust_finish: success to adjust the positon
Turtle_reach_human1
[INFO] [1587385403.151760, 268.757000]: Turtle_reach_human1
```

```
/usr/bin/python: No module named virtualenvwrapper
virtualenvwrapper.sh: There was a problem running the initialization hooks.

If Python could not import the module virtualenvwrapper.hook_loader,
check that virtualenvwrapper has been installed for
VIRTUALENVWRAPPER_PYTHON=/usr/bin/python and that PATH is
set properly.
^source: command not found
^source: command not found
midora@midora-u16:~$ workon HRI
/usr/bin/python: No module named virtualenvwrapper
/usr/bin/python: No module named virtualenvwrapper
(HRI) midora@midora-u16:~$ cd Desktop/pysrc/Human-Robot-Interaction/
(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction$ python tcpNode.
py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Wait for Connection....................
Accept connection from ('45.135.186.119', 45478)
'Human_detect.txt' has been sent to (45.135.186.119:45478)
Wait for Connection....................
```

```
(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction$
python objNode.py
Loading network.....
Network successfully loaded
waiting for object detect message
[INFO] [1587382922.851391, 847.481000]: Object_detected
^C(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction
python objNode.py
Loading network.....
Network successfully loaded
waiting for object detect message
^C(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction
(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction$
python objNode.py
Loading network.....
Network successfully loaded
waiting for object detect message
[INFO] [1587384565.978684, 852.563000]: Object_detected
^C(HRI) midora@midora-u16:~/Desktop/pysrc/Human-Robot-Interaction
python objNode.py
Loading network.....
Network successfully loaded
waiting for object detect message
```

```
0: NEW REQUEST
['my name is John']
Your name is: John
Document sentiment score: 0.0
Language of the text: en
Speaking: Nice to meet you John, what can I do for you?
Listening: Please get me a coffee/water/cookie...

0: NEW REQUEST
```

Figure 18: overall module testing

5 Analysis and Summary

5.1 World file

- **Problem** After improving the simulation environment, save it as a world file using Ctrl+S, but the content of the file is difficult to change.
- **Resolvent** Use another file form.

5.2 Mapping

- **Problem** In the process of map building, because of the single environmental characteristics, the strategy of keyboard controlling the movement of turtlebot, and the limitation of 2D using laser, the map building is poor.
- **Resolvent** Add objects with larger bottom area. In the same time, move slowly when building the map, and make some restrictions on the order of entering each room

5.3 Human simulation

- **Problem** At the beginning of simulation, we used the human simulation of gazebo, but its species of human modle are few, and it is difficult to second creation.
- **Resolvent** Use 3D modeling software such as 3d max, blender or maya. I use blender to create a cube shown as follows, but I haven't learned how to build a house yet. But at the teacher's suggestion, I gave up the idea.

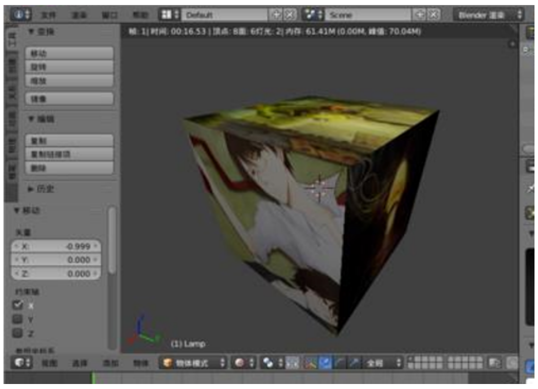


Figure 19: cube in blender

5.4 Access to main process

- **Problem** When the module of simulation is access to the main process, no or too much messages are received by the main process.
- **Resolvent** Set Publisher as a global variable.

5.5 About the project

Every member of our team has completed this project conscientiously, and we often discuss the problems encountered in the group. In general, we have completed this project very well. Of course, the setting of this project itself is very good, which is conducive to exercise our ability in coding, image processing, artificial intelligence, the use of ROS and voice processing. It would be better if we could add physical operation.

6 Appendix

Our code can be obtained from
https://github.com/Mi-Dora/Human-Robot-Interaction.git

References

[1] https://blog.csdn.net/miffy2017may/article/details/97812440

[2] https://blog.csdn.net/qq_27977711/article/details/82024087?utm_source=blogxgwz8

[3] https://wiki.ros.org/

[4] 中国大学 MOOC——《机器人操作系统入门》课程讲义

[5]《ROS 机器人编程》

[6] Programming Robots with ROS