

Part 1 Introduction to URDF Model

1. URDF Model Introduction

The **Unified Robotic Description Format (URDF)** is an **XML file format**, universal description format, **used in ROS to describe the robot structure**.

Robots are usually composed of several **linkages** and **joints** that connect two linkages. **A robot motion model is that multiple linkages are connected and restricted by each other.**

And **URDF files describe the relationship between joints and linkages**, and their **inertial properties, geometric characteristics** and **collision models**.

2. Difference between xacro and URDF Model

The URDF model is a straightforward file for describing robot models, **but its simplicity can become a drawback when describing more complex robots, leading to lengthy description files that lack conciseness.**

In contrast, the xacro model is an extension of the URDF model, and it offers a more advanced way of describing robots. **By using the xacro format, you can simplify the robot description file, improve the reusability of the code, and solve the verbosity problem of the URDF model.**

For instance, when describing the legs of a humanoid robot, using the URDF model would require a separate description of each leg. However, with the xacro model, you can describe a single leg and reuse it throughout the robot's structure.

Overall, the xacro model is a powerful tool that can make robot modeling more efficient, flexible, and maintainable. By leveraging its advanced features, you can create complex robot structures with ease and reduce the overhead associated with manual coding.

3. Install URDF Dependencies

- 1) Input command **"sudo apt update"** and press Enter to update the package information.
- 2) Input command **"sudo apt-get install ros-melodic-urdf"** and press Enter to install URDF model.
- 3) Input command **"sudo apt-get install ros-melodic-xacro"** and press Enter to install URDF model.

4. Basic Syntax of URDF Model

4.1 XML Basic Syntax

As the URDF model is written upon XML standard, you need to get a understanding of the basic structure of XML format.

Element:

You can define the element as **any name**, but you need to adhere to the below format.

```
<element>
</element>
```

Attribute:

Attribute is included in element, which **defines natures and parameters of element**. When using attribute to define element, please conform to the below format.

```
<element  
attribute_1 = "attribute value1"  
attribute_2 = "attribute value2">  
</element>
```

Note: to avoid affecting other attributes and elements, please use the below format to comment.

```
<!-- commented content>
```

4.2 Link

In URDF model, **link** describes the appearance and physical properties of a rigid component. The following tags will be used in editing link action.

```
1 <link name="<link name>">  
2   <inertial>.....</inertial>  
3   <visual>.....</visual>  
4   <collision>.....</collision>  
5 </link>
```

<visual>: describe the **appearance parameters of link**, including **size, color, shape**, etc.

<inertial>: describe **inertial parameters of link** containing **dynamics operation** of the robot.

<collision>: describes the **collision properties of link**.

Each tag has its child tag, and the function of each tag is different. Below is a table of functions of corresponding tags.

Tag	Function
origin	Describe the pose of link. It contains two parameters, including xyz and rpy. xyz: describe the pose of link on simulation map . rpy: describe the posture of link on simulation map .
mass	Describe mass of a link
inertia	Describe inertia of a link . Because of the symmetry of inertial matrix , these six parameters need to be entered as attributes . They can be calculated.
geometry	Describe the shape of a link . It uses mesh parameter to load texture files and uses filename parameter to load the path of texture. Additionally, it contains three child tags, including box , cylinder and sphere .
material	Describe the material of a link . name parameter is a required field. Color and transparency can be adjusted through child tag.

4.3 Joint

In URDF, “**joint**” describes the **properties of joint kinematics and dynamics**, **the position and speed restriction**. According to the types of movement, joints can be divided into 6 types.

Type and Explanation	Tag
Continuous joint : rotate around single axis continuously	continuous
Revolute joint : similar to continuous, but its rotation angle is limited	revolute
Prismatic joint : move along a axis within limited range	prismatic
Planar joints : translate or rotate in plane orthogonal directions	planar
Floating joint : translate and rotate	floating
Fixed joint : not allowed to do any movements	fixed

The following keywords will be adopted to edit joint movements.

```
<joint name="name of joint">  
  <parent link="link1">  
  <child link="link2">  
  <calibration>.....</calibration>  
  <dynamics damping ...../>  
  <limit effort ...../>  
</joint>
```

<parent_link>: parent link

<child_link>: child link

<calibration>: calibrate joint angle

<dynamics>: describe physical attributes of movement

<limit>: describe limit value of movement

Each tag has its child tag, and the function of each tag is different. Below is a table of functions of corresponding tags.

Tag	Function
origin	Describe the pose of link. It contains two parameters, including xyz and rpy. xyz: describe the pose of link on simulation map. rpy: describe the posture of link on simulation map.
axis	Used to control child link to rotate around X, Y or Z axis of parent link
limit	It is used to limit child link. Attributes of lower and upper limit rotation range. effort(unit: N) attribute restricts the force range during rotation. velocity attribute limits rotation speed(unit:m/s)
mimic	Describe the relationship between this joint and other joints.
safety_controller	Describe the parameter of safety controller which can protect joint motion of robot.

4.4 robot Tag

To make top tag of robot complete, `<link>` and `<joint>` tags must be involved in `<robot>`.

```
<robot name="name of robot">  
  <link>.....</link>  
  <link>.....</link>  
  
  <joint>.....</joint>  
  <joint>.....</joint>  
</robot>
```

4.5 gazebo Tag

This tag is used with [gazebo simulator](#) to [set simulation parameters](#). It can be employed to introduce [gazebo plug-ins](#), [physical attributes](#), etc.

```
<gazebo reference="link1">  
  <material>Gazebo/Black</material>  
</gazebo>
```

4.6 Write a Simple URDF Model

```
1 <?xml version="1.0"?>
2 <robot name="pan_tilt">
3
4     <!--定义了base_link -->
5     <!-- <visual>标签描述了在仿真环境的外观, 包括几何外形<geometry> (圆柱形cylinder) 等-->
6     <link name="base_link">
7         <visual>
8             <geometry>
9                 <cylinder length="0.01" radius="0.2" />
10            </geometry>
11            <origin rpy="0 0 0" xyz="0 0 0" />
12            <material name="yellow">
13                <color rgba="1 1 0 1" />
14            </material>
15        </visual>
16    </link>
17
18    <!-- 定义了关节pan_joint, 以及其关节类型: 旋转副 (有限制) -->
19    <!-- 旋转副连接的两个刚体分别为base_link和pan_link -->
20    <joint name="pan_joint" type="revolute">
21        <parent link="base_link" />
22        <child link="pan_link" />
23        <origin xyz="0 0 0.1" />
24        <axis xyz="0 0 1" />
25        <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14" />
26        <dynamics damping="50" friction="1" />
27    </joint>
28
29    <link name="pan_link">
30        <visual>
31            <geometry>
32                <cylinder length="0.4" radius="0.04" />
33            </geometry>
34            <origin rpy="0 0 0" xyz="0 0 0.09" />
35            <material name="red">
36                <color rgba="0 0 1 1" />
37            </material>
38        </visual>
39    </link>
40
41    <joint name="tilt_joint" type="continuous">
42        <parent link="pan_link" />
43        <child link="tilt_link" />
44        <origin xyz="0 0 0.2" />
45        <axis xyz="0 1 0" />
46        <limit effort="300" velocity="0.1" lower="-4.64" upper="-1.5"/>
47        <dynamics damping="50" friction="1"/>
48    </joint>
49
50    <link name="tilt_link">
51        <visual>
52            <geometry>
53                <cylinder length="0.4" radius="0.04" />
54            </geometry>
55            <origin rpy="0 1.5 0" xyz="0 0 0" />
56            <material name="green">
57                <color rgba="1 0 0 1" />
58            </material>
59        </visual>
60    </link>
61
62 </robot>
```

Name the Robot Model

Before writing an URDF model, you need to name the model in this format “<robot name= “ABCD” >”. At last, enter “</robot>” to indicate that robot model is written successfully.

```
1  <?xml version="1.0"?>
2  <robot name="pan_tilt">
```

```
62 </robot>
```

Set a Link

1. When writing the first link, indent to sign that this link belongs to the model. Then name this link in this format “<link name= “name” >”. In the end, enter “</link>” to show that you complete writing the link.

```
6  <link name="base_link">
```

```
16 </link>
```

2. When [writing a link description](#), use indentation to indicate that the description is related to the link being set. You need to enter “<visual>” to at the beginning and “</visual>” at the end.

```
7  <visual>
```

```
15 </visual>
```

3. “<geometry>” [describes the shape of a link](#). To end description, input “</geometry>”. Besides, you need to indent to indicate that the indented content is the description of the shape of a link. For instance, the following content is the description of the shape of a link. “length=“0.01”” means that the length of this link is 0.01m, and “radius=“0.2”” indicates that the radius of this link is 0.2m.

```
8  <geometry>
9  <cylinder length="0.01" radius="0.2" />
10 </geometry>
```

4. “<origin>” [describes the position of a link](#). You need to indent to show the description of link position. “<origin rpy=“0 0 0” xyz=“0 0 0” />” describes the position of a link. rpy is link angle and xyz is the coordinate of the link. “xyz=“0 0 0”” means that the link is at the origin of the coordinate system.

```
11 <origin rpy="0 0 0" xyz="0 0 0" />
```

5. “<material>” [describes the appearance of a link](#). Use indentation to point out the description of the color of a link. “<material>” needs to be input at the beginning of the description and “</material>” at the end. The below codes sets the color of a link as yellow. “<color rgba=“1 1 0 1” />”, rgba=“1 1 0 1”” is the color threshold value.

```
12 <material name="yellow">
13 <color rgba="1 1 0 1" />
14 </material>
```

Set a Joint

1. When writing the first link, indent to sign that this link belongs to the model being set. Then **name this link and set its type in this format** “<joint name=“ABCD” type=“ABCD”>”. In the end, enter “</joint>” to show that you complete writing the link. To learn more about the type of joints, please refer to “4.2 Joint”.

```
20 | <joint name="pan_joint" type="revolute">
```

```
27 | </joint>
```

2. When **writing joint description**, use indent to indicate this description is for the joint being set. In addition, parameters of parent and child need to be set in these format “<parent link= “ABCD” />” and “<child link= “ABCD” />”. Joint will take the parent link as fulcrum to rotate the child link.

```
21 | <parent link="base_link" />
```

```
22 | <child link="pan_link" />
```

3. “<origin>” **describes the position of a joint**. You need to use indentation to show the description for link position. “<origin xyz= “0 0 0.1” />” describes the position of a link. xyz is the coordinate of a joint.

```
23 | <origin xyz="0 0 0.1" />
```

4. “<axis>” **describes the position of a link**. Use indentation to indicate that the description is related to the link position. The below content “<axis xyz= “0 0 1” />” describes the posture of a joint. xyz is posture position of the joint.

```
24 | <axis xyz="0 0 1" />
```

5. “<limit>” **is used to limit the motion of a joint**. When describing a joint, you need to use indentation to this description is related to limitation of joint angle. The below content describes that the force used to rotate joint should not exceed 300N, the upper limit of rotation range is 3.14 radian and lower limit of rotation range is -3.14 radian. You can them in these format “effort=“joint force(N)”、velocity=“rotation speed of a joint”、lower=“lower limit of rotation range”、upper=“upper limit of rotation range””.


```
25 | <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14" />
```

6. “<dynamics>” **describes dynamics of a joint**. When describing dynamics of a joint, use indentation to indicate this description is related to joint posture. “<dynamics damping=“50” friction=“1” />” describes dynamics parameters.

```
26 | <dynamics damping="50" friction="1" />
```

Part 2 ROS Robot URDF Model Description

1) Start JetAuto, then connect it to NoMachine.

2) Double click  to open the command line terminal

3) Input command “**roscd jetauto_description/urdf/**”, then press Enter to enter the startup program directory.

- 4) Input command “**vim jetauto_car.gazebo.xacro**” to open JetAuto simulation model file.
5) Locate the following codes.

```
20 <xacro:model_color link_name="base_link" color_name="green"/>
21 <xacro:model_color link_name="lidar_link" color_name="black"/>
22 <xacro:model_color link_name="back_shell_link" color_name="black"/>
23 <xacro:model_color link_name="wheel_right_front_link" color_name="black"/>
24 <xacro:model_color link_name="wheel_left_front_link" color_name="black"/>
25 <xacro:model_color link_name="wheel_right_back_link" color_name="black"/>
26 <xacro:model_color link_name="wheel_left_back_link" color_name="black"/>
27 <xacro:model_color link_name="mic_link" color_name="black"/>
28 <xacro:model_color link_name="speaker_link" color_name="black"/>
```

These codes describe each component composing the robot. “link_name” is the name of the robot component, and “color_name” represents the color of the robot component.

“base_link” refers to the main body of JetAuto

“lidar_link” is the simulation model of Lidar

“back_shell_link” is the simulation model of back shell of the controller

“wheel_right_front_link” is the simulation model of the front right Mecanum wheel

“wheel_left_front_link” is the simulation model of the front left Mecanum wheel

“wheel_right_back_link” is the simulation model of right hind Mecanum wheel

“wheel_left_back_link” is the simulation model of left hind Mecanum wheel

“mic_link” is the simulation model of Microphone array

“speaker_link” is the simulation model of the bottom speaker

3. JetAuto Model Introduction

Open a new terminal, then enter command “**vim jetauto_car.urdf.xacro**” to open JetAuto model file which contains descriptions for various parts of robot.

3.1 Basic Parameters of Model of JetAuto’s body

“M_PI” defines the pi of model of JetAuto’s body.

“base_link_mass” defines the mass of model of JetAuto’s body.

“base_link_w” defines the width of model of JetAuto’s body.

“base_link_h” defines the height of model of JetAuto’s body.

“base_link_d” defines the length of model of JetAuto’s body.

“wheel_link_mass” defines the mass of model of JetAuto’s mecanum wheel

“wheel_link_radius” defines the radius of model of JetAuto’s mecanum wheel

“wheel_link_length” defines the length of model of JetAuto’s mecanum wheel

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <robot name="jetauto" xmlns:xacro="http://ros.org/wiki/xacro" >
3   <xacro:property name="M_PI" value="3.1415926535897931"/>
4   <xacro:property name="base_link_mass" value="1.6" />.
5   <xacro:property name="base_link_w" value="0.297"/>
6   <xacro:property name="base_link_h" value="0.145"/>
7   <xacro:property name="base_link_d" value="0.11255"/>
8
9   <xacro:property name="wheel_link_mass" value="0.1" />
10  <xacro:property name="wheel_link_radius" value="0.049"/>
11  <xacro:property name="wheel_link_length" value="0.04167"/>
12
```

3.2 Shaded Part of JetAuto Model(base_footpring)

This part defines link “base_footpring” and joint “base_joint”. Joint “base_joint” can be used to connect the link “base_footpring” and the link “base_link”. Its position is defined at the origin to avoid defined mecanum wheels from being embedded into the ground.


```

13 <link name="base_footprint"/>
14
15 <joint name="base_joint" type="fixed">
16   <parent link="base_footprint"/>
17   <child link="base_link"/>
18   <origin xyz="0.0 0.0 0.0" rpy="0 0 0"/>
19 </joint>

```

3.3 Model of JetAuto's Main Body(base_link)

This part defines the link **"base_link"** as a rectangle and the parameters defined previously have been filled in.

To define the visual appearance of the "base_link", use the "<visual>" tag and include the position of the origin using the "origin" keyword.

Import the relevant dynamics profile using the "geometry" keyword and define the color as green using the "material" keyword.

Next, define the collision behavior of the "base_link" using the "<collision>" tag. Specify the location of the collision using the "origin" keyword and define the collision area using the "geometry" keyword.

```

21 <link
22   name="base_link">
23   <xacro:box_inertial m="${base_link_mass}" w="${base_link_w}" h="${base_link_h}" d="${base_link_d}"/>
24   <visual>
25     <origin
26       xyz="0 0 0"
27       rpy="0 0 0" />
28     <geometry>
29       <mesh
30         filename="package://jetauto_description/meshes/base_link.stl" />
31     </geometry>
32     <material name="green"/>
33   </visual>
34   <collision>
35     <origin
36       xyz="${base_link_w/2.0 - 0.14810} 0 ${0.126437/2 + 0.02362364}"
37       rpy="0 0 0" />
38     <geometry>
39       <box size="${base_link_w} ${base_link_h} ${base_link_d}" />
40     </geometry>
41   </collision>
42 </link>

```

3.4 Simulation Model of JetAuto's Back Shell(back_shell_link)

This part defines the link **"back_shell_link"** as a rectangle and the parameters defined previously have been filled in.

To define inertia of the "back_shell_link", use the "<inertial>" tag and include the origin of inertia using the "origin" keyword, mass using keyword "mass" and inertial properties of links using keyword **"inertia"**.

To define the visual appearance of the "back_shell_link", use the "<visual>" tag and include the position of the origin using the "origin" keyword.

Import the relevant dynamics profile using the "geometry" keyword and define the color as black using the "material" keyword.

Next, define the collision behavior of the "back_shell_link" using the "<collision>" tag. Specify the location of the collision using the "origin" keyword and define the collision area using the "geometry" keyword.

```

42 </link>
43 <link
44   name="back_shell_link">
45   <inertial>
46     <origin
47       xyz="-1.22838595456587E-05 0.00218574826309681 -0.0500522861933898"
48       rpy="0 0 0" />
49     <mass
50       value="0.0663478534899862" />
51     <inertia
52       ixx="5.65277934912267E-05"
53       ixy="-5.13394387877366E-11"
54       ixz="-4.07561372273553E-11"
55       iyy="4.33740893441632E-05"
56       iyz="-5.43059341238134E-06"
57       izz="6.86642544694324E-05" />
58   </inertial>
59   <visual>
60     <origin
61       xyz="0 0 0"
62       rpy="0 0 0" />
63     <geometry>
64       <mesh
65         filename="package://jetauto_description/meshes/back_shell_link.stl" />
66       </geometry>
67       <material name="black">
68       </material>
69     </visual>
70     <collision>
71       <origin
72         xyz="0 0 0"
73         rpy="0 0 0" />
74       <geometry>
75         <mesh
76           filename="package://jetauto_description/meshes/back_shell_link.stl" />
77         </geometry>
78       </collision>
79 </link>

```

3.5 Simulation Model of JetAuto Mecanum Wheel(Take Right Front Wheel as Example)

Define the link "**wheel_right_front_link**", then define the link "**wheel_right_link**" as a cylinder and fill in the parameters defined previously.

To define the visual appearance of the "**wheel_right_front_link**", use the "<visual>" tag and include the position of the origin using the "origin" keyword.

Import the relevant dynamics profile using the "geometry" keyword and define the color as black using the "material" keyword.

Next, define the collision behavior of the "**wheel_right_front_link**" using the "<collision>" tag. Specify the location of the collision using the "origin" keyword and define the collision area using the "geometry" keyword.

```

93 <link
94   name="wheel_right_front_link">
95   <xacro:cylinder_inertial m="${wheel_link_mass}" r="${wheel_link_radius}" h="${wheel_link_length}" />
96   <visual>
97     <origin
98       xyz="0 0 0"
99       rpy="0 0 0" />
100   <geometry>
101     <mesh
102       filename="package://jetauto_description/meshes/wheel_right_front_link.stl" />
103     </geometry>
104     <material name="black">
105     </material>
106   </visual>
107   <collision>
108     <origin
109       xyz="0 0 ${wheel_link_length/2.0}"
110       rpy="0 0 0" />
111     <geometry>
112       <cylinder length="${wheel_link_length}" radius="${wheel_link_radius}" />
113     </geometry>
114   </collision>
115 </link>

```

3.6 Simulation Model of JetAuto's Microphone Array

Define the link "**mic_link**".

To define inertia of the "**mic_link**", use the "<inertial>" tag and include the origin of inertia using the "origin" keyword, mass using keyword "mass" and inertial properties of links using keyword "**inertia**".

To define the visual appearance of the "**mic_link**", use the "<visual>" tag and include the position of the origin using the "origin" keyword.

Import the relevant dynamics profile using the "geometry" keyword and define the color as black using the "material" keyword.

Next, define the collision behavior of the "**mic_link**" using the "<collision>" tag. Specify the location of the collision using the "origin" keyword and define the collision area using the "geometry" keyword.

```
225 <link
226   name="mic_link">
227   <inertial>
228     <origin
229       xyz="7.37972827624667E-07 0.000380767498086923 -0.0010743560446495"
230       rpy="0 0 0" />
231     <mass
232       value="0.0234232570822793" />
233     <inertia
234       ixx="1.59705800233789E-05"
235       ixy="-5.64325349754072E-11"
236       ixz="8.96027229707658E-11"
237       iyy="2.23652963143563E-05"
238       iyz="6.01204669252721E-08"
239       izz="3.74624298483869E-05" />
240   </inertial>
241   <visual>
242     <origin
243       xyz="0 0 0"
244       rpy="0 0 0" />
245     <geometry>
246       <mesh
247         filename="package://jetauto_description/meshes/mic_link.stl" />
248       </geometry>
249       <material name="black"/>
250   </visual>
251   <collision>
252     <origin
253       xyz="0 0 0"
254       rpy="0 0 0" />
255     <geometry>
256       <mesh
257         filename="package://jetauto_description/meshes/mic_link.stl" />
258       </geometry>
259   </collision>
260 </link>
```

3.7 Simulation Model of JetAuto's Speaker(speaker_link)

Define the link "**speaker_link**".

To define inertia of the "**speaker_link**", use the "<inertial>" tag and include the origin of inertia using the "origin" keyword, mass using keyword "mass" and inertial properties of links using keyword "**inertia**".

To define the visual appearance of the "**speaker_link**", use the "<visual>" tag and include the position of the origin using the "origin" keyword.

Import the relevant dynamics profile using the "geometry" keyword and define the color as black using the "material" keyword.

Next, define the collision behavior of the "**speaker_link**" using the "<collision>" tag. Specify the location of the collision using the "origin" keyword and define the collision area using the "geometry" keyword.

```
274 <link
275   name="speaker_link">
276   <inertial>
277     <origin
278       xyz="0 0 0"
279       rpy="0 0 0" />
280     <mass
281       value="0" />
282     <inertia
283       ixx="0"
284       ixy="0"
285       ixz="0"
286       iyy="0"
287       iyz="0"
288       izz="0" />
289   </inertial>
290   <visual>
291     <origin
292       xyz="0 0 0"
293       rpy="0 0 0" />
294     <geometry>
295       <mesh
296         filename="package://jetauto_description/meshes/speaker_link.stl" />
297     </geometry>
298     <material name="black"/>
299   </visual>
300   <collision>
301     <origin
302       xyz="0 0 0"
303       rpy="0 0 0" />
304     <geometry>
305       <mesh
306         filename="package://jetauto_description/meshes/speaker_link.stl" />
307     </geometry>
308   </collision>
309 </link>
```

Part 3 Introduction to Gazebo

1. Gazebo Description

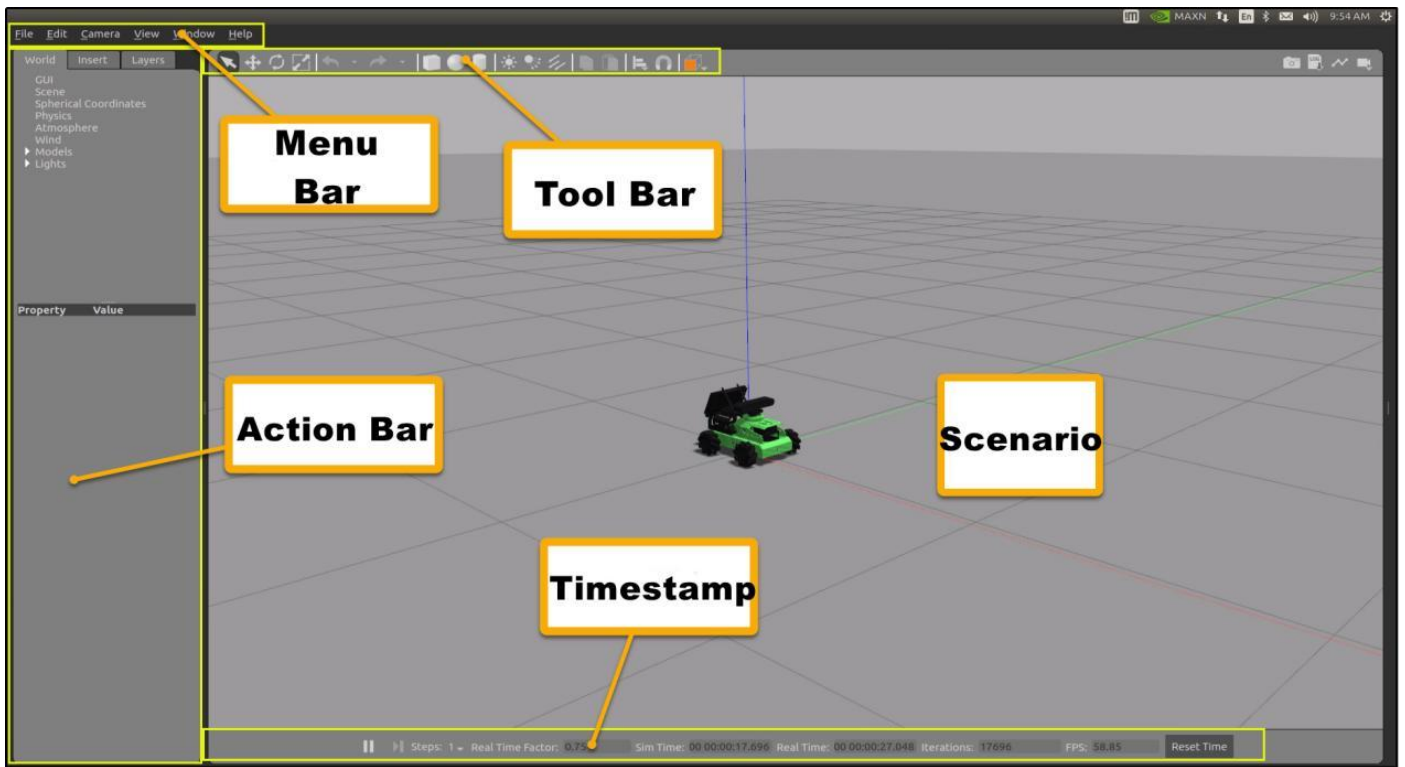
On simulation software, a realistic virtual physical environment can be rendered to test and experiment how the robot complete the task.

Gazebo is the most commonly used robot simulation software in the ROS system. It can provide high-fidelity physical simulation conditions and a complete set of sensor models. It offers friendly interaction between the simulation environment and the robot, and can improve the robot's performance in complex environments.

Gazebo supports files in urdf and sdf format which are adopted to describe simulation environment. And the official provides various commonly used models that can be directly used.

2. Introduction to Gazebo GUI

The Gazebo GUI is as follow.



The function of each part is listed below.


Name	Function
Menu bar	Configure or modify the parameters of the simulation software and some interactive functions
Tool bar	Provide the most commonly used options for interaction with simulator
Timestamp	Set the time of the virtual space
Action bar	Make any operations on the models and modify the parameters
Scenario	The main part of the simulator where the simulated model is displayed

Part 4 Gazebo Xacro Model Visualization

1. Start Simulation

To get a clear understanding of JetAuto's model and structure, you can visualize the robot model through Gazebo.

1) Start JetAuto, then connect to ubuntu desktop through NoMachine.

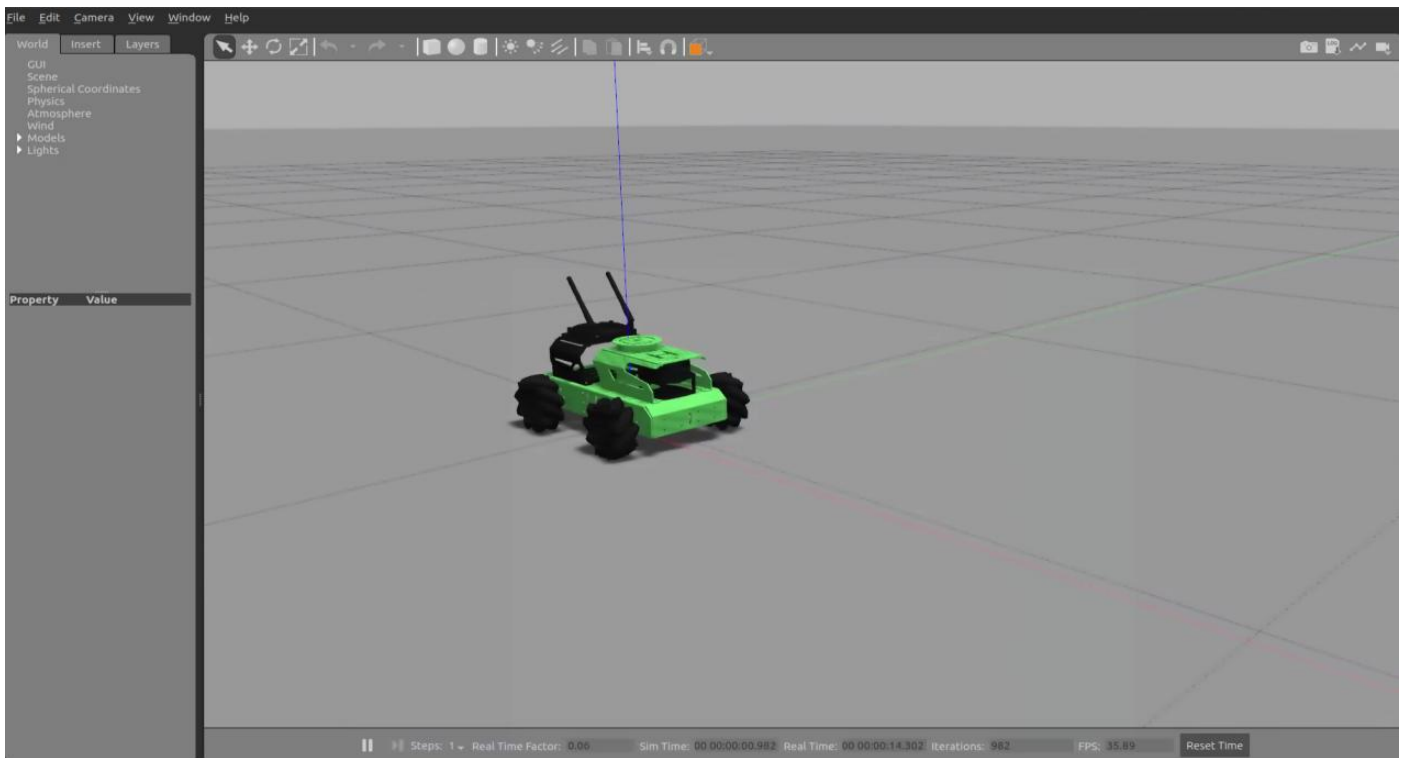
2) Then click  to open command line terminal.

3) Input command "**sudo systemctl stop start_app_node.service**" and press Enter to close APP service.

4) Input command "**roslaunch jetauto_gazebo worlds.launch**" to open JetAuto simulation model.

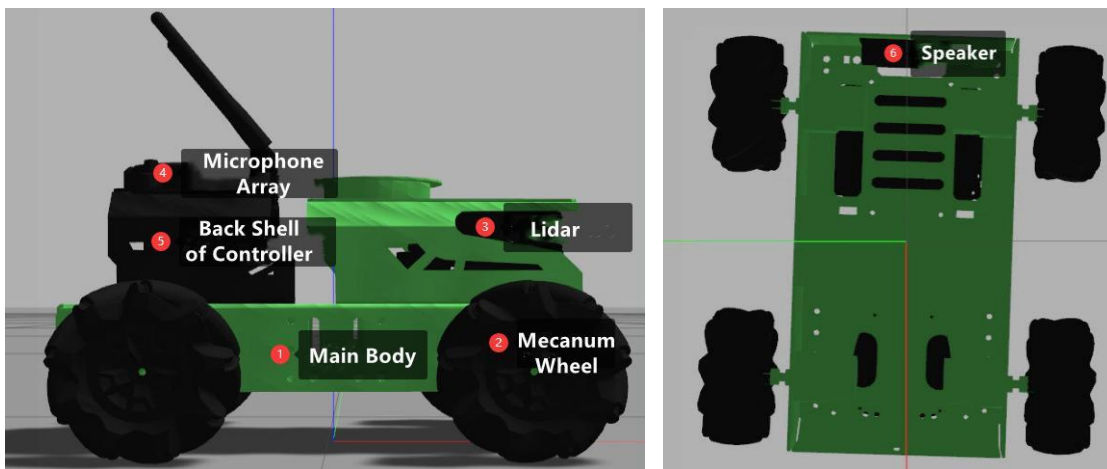
And the simulation model is as below.

5) Enter the command "**sudo systemctl start start_app_node.service**" to restart the mobile app service and wait for a beep sound to indicate that the service has started successfully.



2. Introduction to Robot Model

JetAuto consists of main body, four mecanum wheels, controller, speaker, Lidar and microphone array.




Part 5 Gazebo Hardware Simulation

To get a clear understanding of simulation models of various expanded devices, you need to master related codes of model.

1. Depth Camera Simulation

1) Start JetAuto, then connect to ubuntu desktop through NoMachine.

2) Then click  to open command line terminal.

3) Input command “**roscd jetauto_description/urdf/**” and press Enter to enter the startup program directory.

4) Input command **"vim depth_camera.gazebo.xacro"** to open depth camera simulation file.

This file describes the **name, size, position, display settings, topics and other properties of the depth camera simulation model**. To understand this file, you need to learn about related grammar referring to **"7. Motion Control Part/2. Simulation Model and URDF/Part 1 Introduction to URDF Model"**.

2. Lidar Simulation

1) Start JetAuto, then connect to ubuntu desktop through NoMachine.

2) Then click  to open command line terminal.

3) Input command **"roscd jetauto_description/urdf/"** and press Enter to enter the startup program directory.

4) Input command **"vim lidar.gazebo.xacro"** to open Lidar simulation model file.

This file describes **the name, detection range, position, noise-canceling settings, topics and other properties of the Lidar simulation model**. To understand this file, you can learn about related grammar referring to **"7. Motion Control Part/2. Simulation Model and URDF/Part 1 Introduction to URDF Model"**.

3. IMU Simulation

1) Start JetAuto, then connect to ubuntu desktop through NoMachine.

2) Then click  to open command line terminal.

3) Input command **"roscd jetauto_description/urdf/"** and press Enter to enter the startup program directory.


4) Input command **"vim imu.gazebo.xacro"** to open Lidar simulation model file.

This file describes **the name, offset settings, noise-canceling settings, acceleration settings, topics and other properties of IMU simulation model**. To understand this file, you can learn about related grammar referring to **"7. Motion Control Part/2. Simulation Model and URDF/Part 1 Introduction to URDF Model"**.

Part 6 Gazebo Mapping Simulation

If you need to create a scenario that is ideal and cannot be replicated in the real world, Gazebo provides a platform to build and map such an environment.

1) Start JetAuto, then connect to ubuntu desktop through NoMachine.

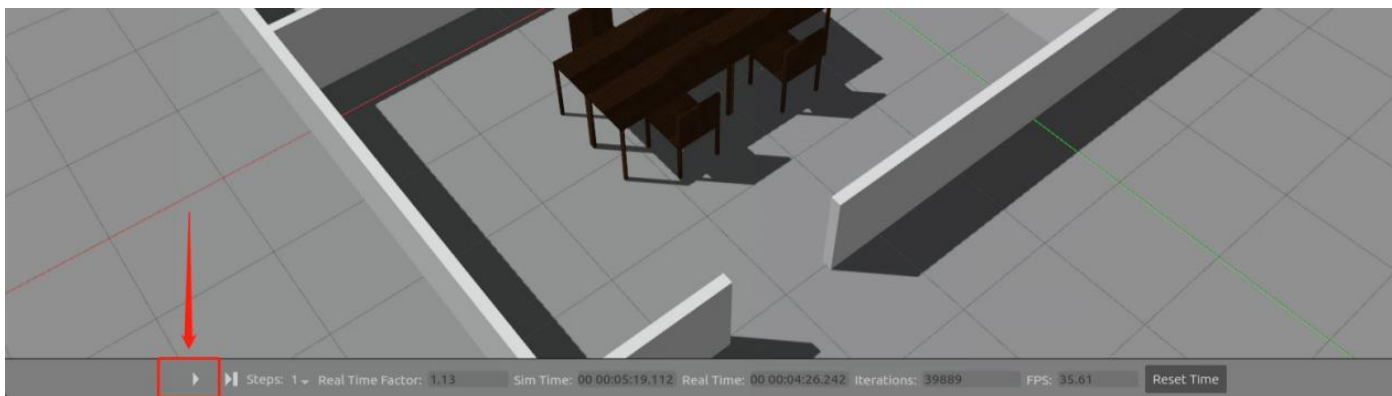
2) Then click  to open command line terminal.

3) Input command **"sudo systemctl stop start_app_node.service"** and press Enter to enter the startup program directory.

4) Input command **"roscd jetauto_gazebo/launch/"** to enter the simulation program directory.

5) Input command **"roslaunch room_worlds.launch"** to start the simulation program.

6) Click the button shown below to start simulation.

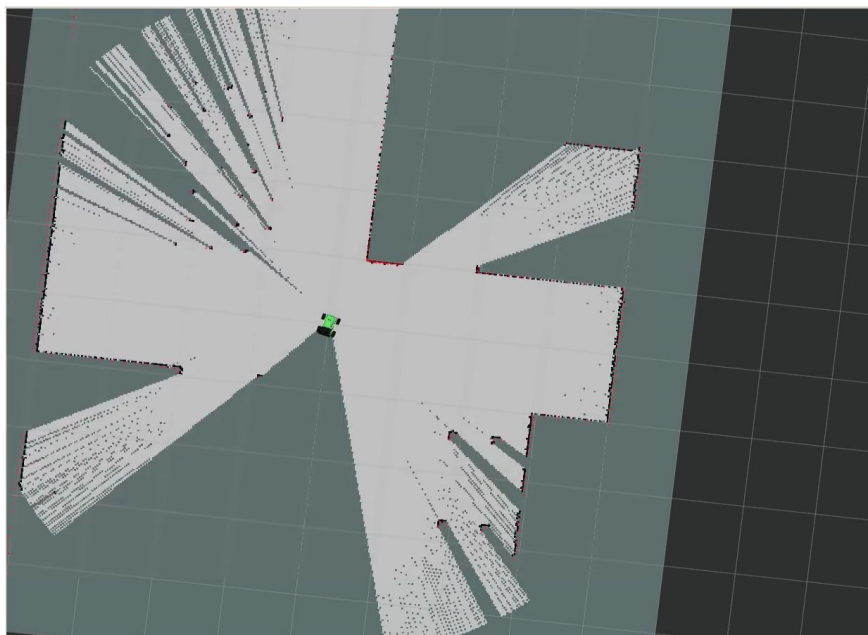


7) Open a new terminal, then input command “**roslaunch jetauto_slam jetauto_robot.launch sim:=true**” to enable basic Lidar service.

8) Open a new command line terminal and input the command “**roslaunch jetauto_slam slam.launch slam_methods:=gmapping**” to enable mapping service.

9) Open a new command line terminal, and input command “**roslaunch jetauto_slam rviz_slam.launch sim:=true slam_methods:=gmapping**” to open RVIZ.

After entering RVIZ, the interface is as follow.

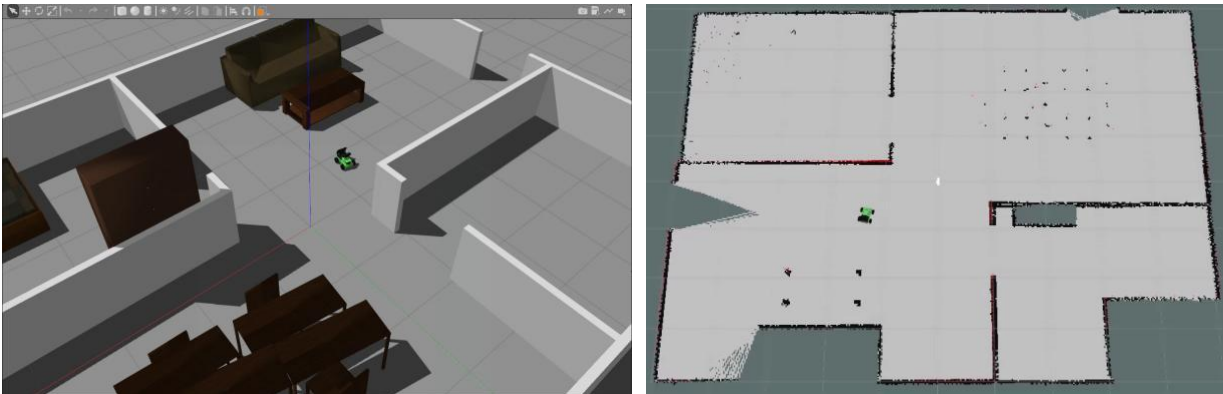


10) Turn on your handle, and use it to control robot move on the simulated map. And the functions of buttons on the handle are listed as below.

Button	Function	Usage
START	stop and reset the robot	Short press
Move the left joystick upward	move forward	Long press
Move the left joystick downward	move backward	Long press

Move the left joystick left	move left	Long press
Move the left joystick right	move right	Long press
Move the right joystick left	turn left	Long press
Move the right joystick right	turn right	Long press

11) As the robot moves, RVIZ will display the mapping result simultaneously



12) After robot finishes mapping the surroundings, input command “**roscd jetauto_slam/maps**” to enter the directory where the map is saved.

13) Input command “**roslaunch map_server map_saver -f map_01 map:=/jetauto_1/map**” to save the map.

“**map_01**” is the name of the map, and you rename it. If the following hints pop up, the map is saved successfully.

```
[ INFO] [1660362401.992709256]: Waiting for the map
[ INFO] [1660362402.248269725]: Received a 832 X 384 map @ 0.025 m/pix
[ INFO] [1660362402.248373631]: Writing map occupancy data to map_01.pgm
[ INFO] [1660362402.279224621]: Writing map occupancy data to map_01.yaml
[ INFO] [1660362402.279653944]: Done
```

14) Press “**Ctrl+C**” in the terminal, and you can close the current program.

Part 7 Gazebo Navigation Simulation

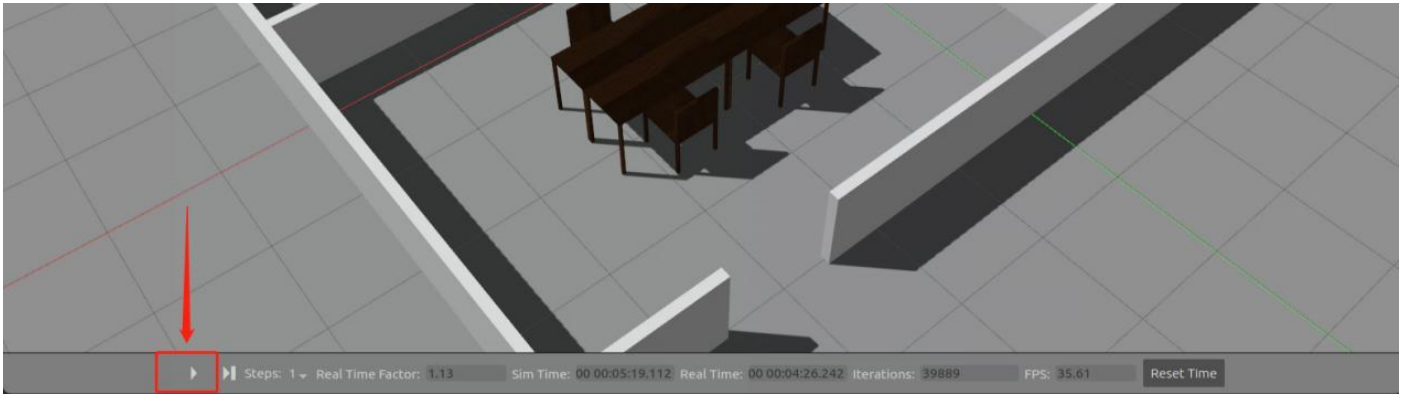
1. Configuration

When we complete the build with the Gazebo software, again we can we navigate within the scene of the Gazebo software to achieve the desired experimental effect.

1) Start JetAuto, and connect it to NoMachine.

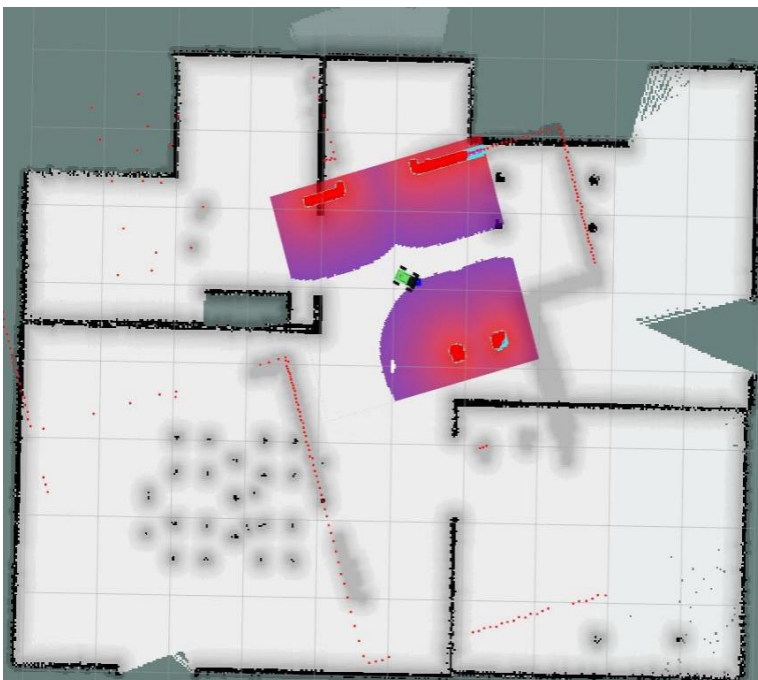
2) Double click to open the command line terminal.

- 3) Input command **"sudo systemctl stop start_app_node.service"** and press Enter to enter the startup program directory.
- 4) Input command **"roscd jetauto_gazebo/launch/"** to enter the simulation program directory
- 5) Input command **"roslaunch room_worlds.launch"** to start the simulation program.
- 6) Click the button shown below to start simulation.



- 7) Open a new terminal, then input the command **"roslaunch jetauto_slam jetauto_robot.launch sim:=true"** to enable basic Liadr service.
 - 8) Open a new terminal, then input command **"roslaunch jetauto_navigation load_map.launch map:=map_01"** to import the map for navigation.

"map_01" is the name of the map, and you can rename it. The map is saved in **"/home/jetauto_ws/src/jetauto_slam/maps"**.
 - 9) Open a new command line terminal, and input the command **"roslaunch jetauto_navigation navigation.launch"** to enable the navigation service.
 - 10) Open a new terminal, and input command **"roslaunch jetauto_navigation rviz_navigation.launch"** to open RVIZ.
- After entering RVIZ, the navigation image is as follow.



2. Start Navigation

Three tools are mainly used in navigation, including **2D Pose Estimate**, **2D Nav Goal** and **Publish Point**.

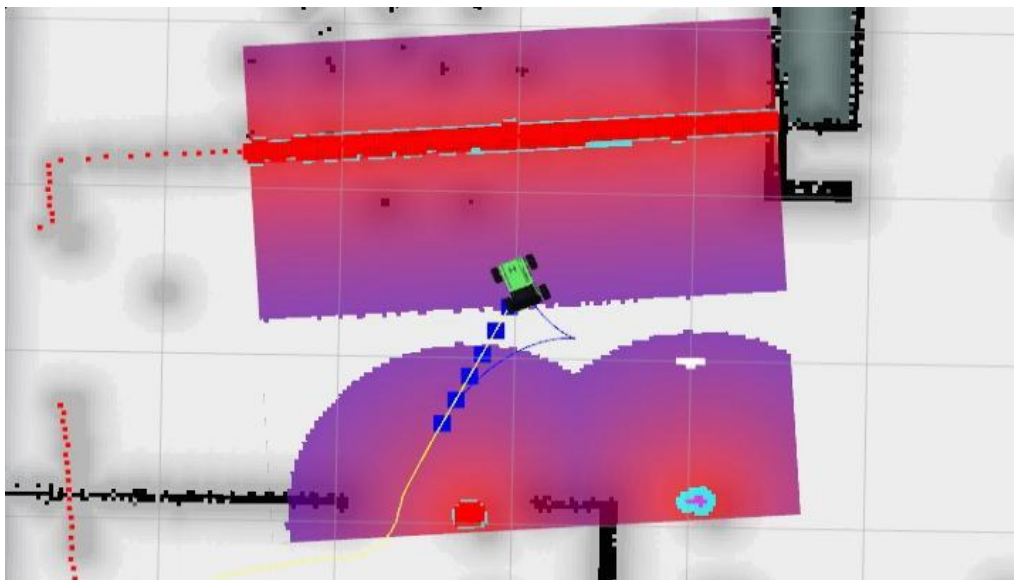


2D Pose Estimate is used to set the initial position of the robot, and **2D Nav Goal** is used to set single target point position, and **Publish Point** is used to set multiple target point position.

Click **2D Nav Goal**, then randomly select one point as the destination by clicking the mouse. After the destination is set, JetAuto will design a path automatically and move to the target point.



Two paths will be generated after you set the target position. The line connecting the blue blocks refers to the direct path between the robot and the target point. And the dark blue line represents the path planned by the robot.



Note: if you need to pause the navigation, you can click **2D Nav Goal**, and then click the robot on the image to set the current position as the target point.