User manual for Agilex Cobot kit

User manual for Agilex Cobot kit

0 Introduction

Agilex Cobot series development kit is developed specifically for the industry research and education application. This development kit is based on Agilex's ROS product ecosystem，which highly meet the flexible requirement on research and education robots by integrating high performance industrial computer, high accuracy LiDAR，multiple sensors，multi-DOF robot arm and visual perception. It includes functions of multi-line lidar automatic navigation，robot arm moveit motion control planning，visual identity and arm automatically grab to give a solution and ultimate user experience to customer for high technical complexity and high integration difficulty applications. The product including the whole technical manuals and all codes open-source will reduce the difficulty of user's specific application and study. The development kit can be widely used in agriculture，intelligent manufacturing，education training，scientific discovery and so on.

1 Main configuration list and parameter specification

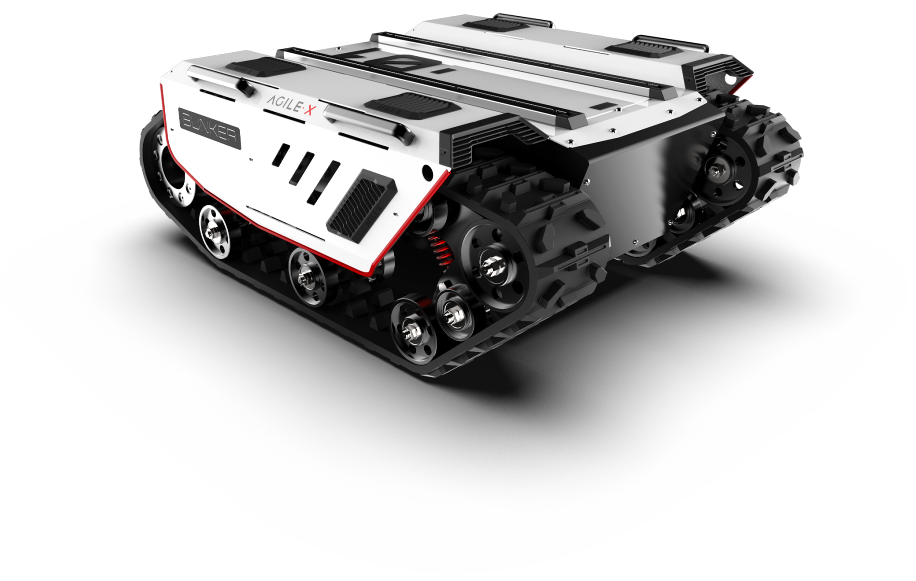
1.1Main configuration

Version	AgileX Pro
Chassis	BUNKER
Robot arm	Aubo i3/5
Robot arm claw	DH AG 95
IPC	x-7010(i7 16G 256SSD)
Visual sensor	RealSense D435
LIDAR	VLP 16
Display	HD screen
Router	B316 4G router

1.2 Main parts

- ### BUNKER introduction

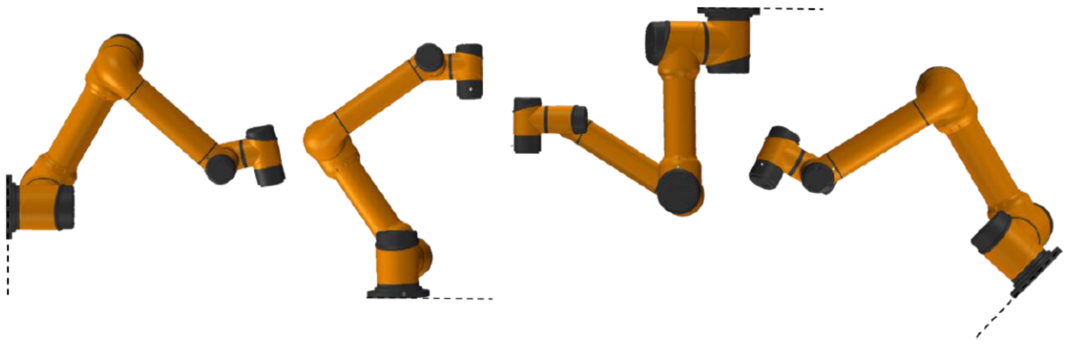
  BUNKER is a versatile industrial application chassis. It's easily and flexibly operated with large development space，adapting variety applications，independent suspension system，large payload capability with suspension，excellent climbing ability including stairs and so on. It can be used for development or solutions of special robots with exploration of inspection，rescue and explosive clearing，special photography and special transportation functions.



Items	Specification
Overall dimension	1000x750x360mm
Internal box dimension	600x300x230mm
Chassis height	90mm
Track width	150mm
Length of ground contact	520mm
Dead weight	About 130kg
Payload	70kg
Power supply	Lithium battery
Battery capacity	30AH
Supply voltage	48V
Climbing ability	⩽35°
Speed	0~1.5m/s
Minimum turning radius	Support situ turning
Obstacle crossing ability	⩽ 170mm
Control mode	Remote control
Remote controller	2.4G /⩽1KM
Communication interface	CAN

- ## Aubo robot arm

  Aubo series robots are a kind of industrial lightweight class collaborative robot which are strictly selected by Agilex according to the research and education industry. It uses joint modular design and system facing developers. Users can develop their own robot control system based on the application program interfaces provided by Aubo platform. Also, Aubo provide specialized programable operation interface and users can observe the robot running status in real time，control setting or off-line simulation. It can greatly improve the work efficiency of practical application. At the same time, we have done ROS adaptation to support Moveit or other open source solutions according to research and education industry.



Robot arm model	**AUBO-i5**
Weight	24 kg
Payload	5 kg
Extend	924 mm
Joint range	-175° ~ +175°
Joint speed	1-3 joint：150°/s 4-6 joint：180°/s
Tool speed	⩽2.8 m/s
Repeated positioning accuracy	± 0.02 mm
Floor space	Ø172 mm
DOF	6
Standard control cabinet dimension（Length*****Height*****Width）	727mm x 623mm x 235mm
I/O power	Control cabinet：24V 3A， Tool： 0V/12V/24V 0.8A
Communication protocol	Ethernet、Modbus - RTU/TCP
Interface	SDK（support C\C++\Lua\Python），support ROS and API
Programming	On the 12.5inch touchscreen. AUBOPE graphical user interface.
Noise	Low
IP level	IP54
Power consumption	About 200W during running

- ## X-7010 IPC introduction

  X-7010 is a modularized and high performance small IPC, customized for high calculation ability and environment requirement of movable robots. It's using Intel platform and support 8/9th 35W high speed processor. It adopts large area aluminum fin of high efficiency heat pipe and active/passive dual heat dissipation design of PWM fan. The strong body of aluminum alloy made by mold ensures its long life and stable operation. It is suitable for intelligent robot, unmanned driving, machine vision, smart city and other fields with high computing requirements. Ubuntu 18.04 and ROS Melodic（Full Desktop) are preloaded on the IPC with some common developing environment for robot development. User can use it directly when just turn on.

  - Palm-sized compact ultra-small body；
  - Support Intel 8th desktop-level high performance CPU；
  - Equipped with heat pipe and intelligent fan for active and passive efficient heat dissipation；
  - Support miniPCIE, NVME and variety of acceleration card expansion scheme；
  - Multi-channel ultra high speed special serial port, suitable for various radar applications；
  - Multi-channel USB3.1 Gen2 and high-speed communication with dual gigabit networks；
  - Solid moulded aluminum alloy body, in line with vehicle vibration and impact；
  - -20~60℃ working temperature；

- **Vision sensor-RealSense D435**

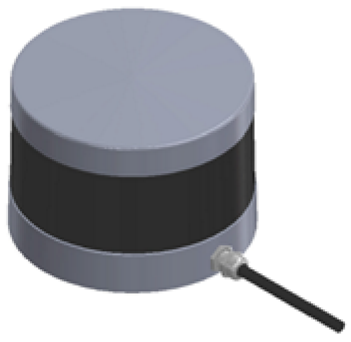
  Binocular vision sensors have a wide range of application scenarios and requirements in robot vision measurement, visual navigation and other robot industry directions. We have selected the common vision sensors in the scientific research and education industry. Equipped with a global image shutter and wide field of view, the Intel Photorealistic Depth Camera D435 effectively captures and streams depth data of moving objects, thus providing highly accurate depth perception for moving prototypes.

	Model	ModelRealsense D435
Basic characteristics	Application scenarios	Indoor/Outdoor
Measuring distance	About 10 meters	
Depth shutter type	Global shutter /3um X 3um	
Support IMU or not	Yes	
Depth camera	Deepin	Active infrared
FOV	86° x 57°（±3°）	
Minimum depth distance	0.105m	
Depth resolution	1280 x 720	
Maximum measuring range	About 10meter	
Depth frame rate	90 fps	
RGB	Resolution	1280 x 800
FOV	69.4° × 42.5°（±3°）	
Frame rate	30fps	
Others	Dimension	90mm x 25mm x 25mm
Interface type	USB-C 3.1	

- ## LIDAR

  The VLP-16 is one of Velodyne's smaller and more advanced lidar products. The VLP-16 is more cost-effective than comparable sensors and retains some of Velodyne's key lidar breakthrough features: real-time, 360°, THREE-DIMENSIONAL coordinates and range, and calibrated reflectivity measurements.

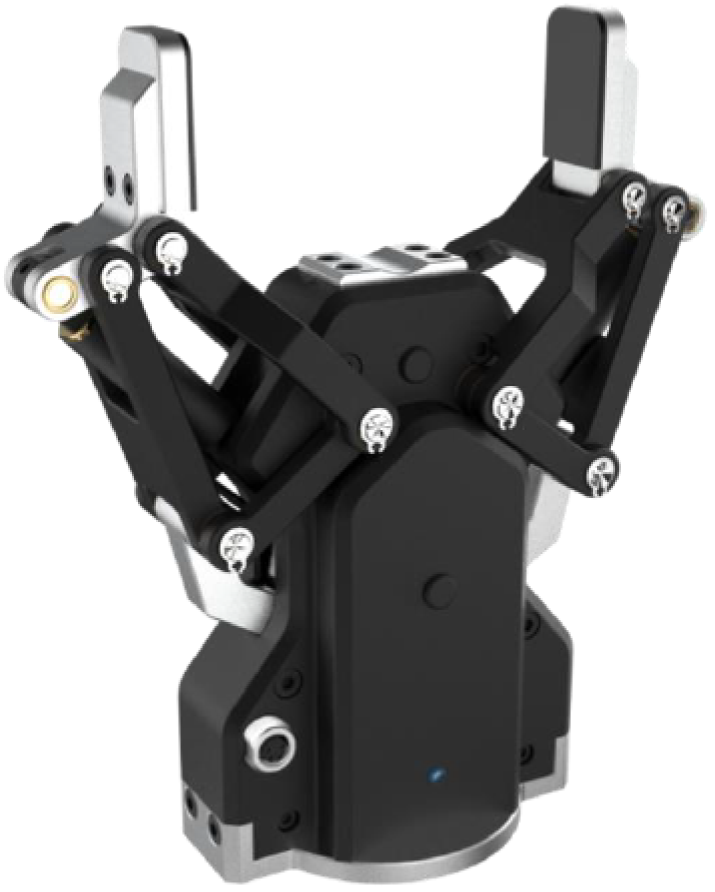
  The VLP-16 has a measuring range of up to 100 m, low power consumption (about 8 W), light weight (about 830 g), small size (ø 103mm x 72mm), and dual return performance, making it ideal for backpack-like measurements, drone mounts, and other mobile devices.



Items	**Parameters**
Maximum range	100m
Range accuracy	±3cm
Scanning rate	One single echo：300,000 points/sec， Dual echo：600,000points/sec
Vertical Viewing Angle	-15°~＋15°
Vertical angle resolution	2°
Scan frequency	5Hz~20Hz
Safety class	Class 1
Weight	830g
Power consumption	8W
Voltage	9V~18V
Temperature	-10℃~＋60℃

- ## Gripper

  DH Robot AG 95 electric gripper has two adaptive parallel mechanical knuckle fingers. Each knuckle finger consists of multiple connecting rod mechanisms and a spring, as shown in FIG. 1.1. Knuckle fingers can make up to five points of contact with an object. Knuckle fingers are driven by underdrive control, making the motor less than the total number of joints. This design simplifies the control mode of grasping, so that the joint fingers can automatically adapt to the shape of the object they grasp.

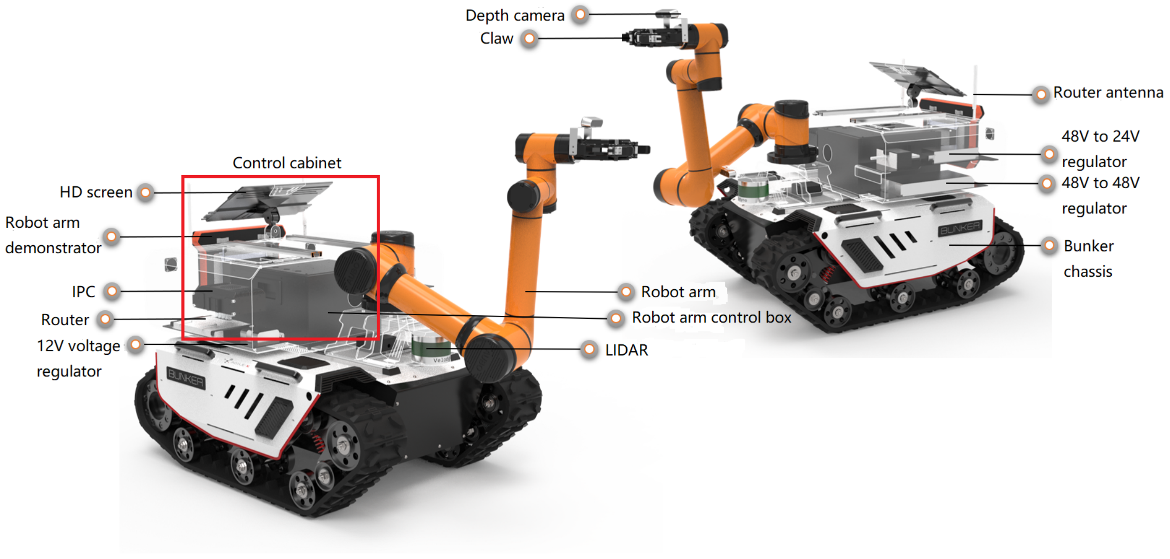


Maximum recommended load	**3-5kg**
Finger opening and closing stroke（Programable）	0-95mm
Grasping force（Programable）	45-160N
Fastest finger opening and closing speed	190mm/s
Dead weight	1kg
Finger repeat positioning accuracy	0.03mm
Communication protocol	TCP/IP, USB2.0, RS485, I/O, CAN2.0A, EtherCAT（Optional）
Working voltage	24V DC±10%
Working temperature	0~50℃

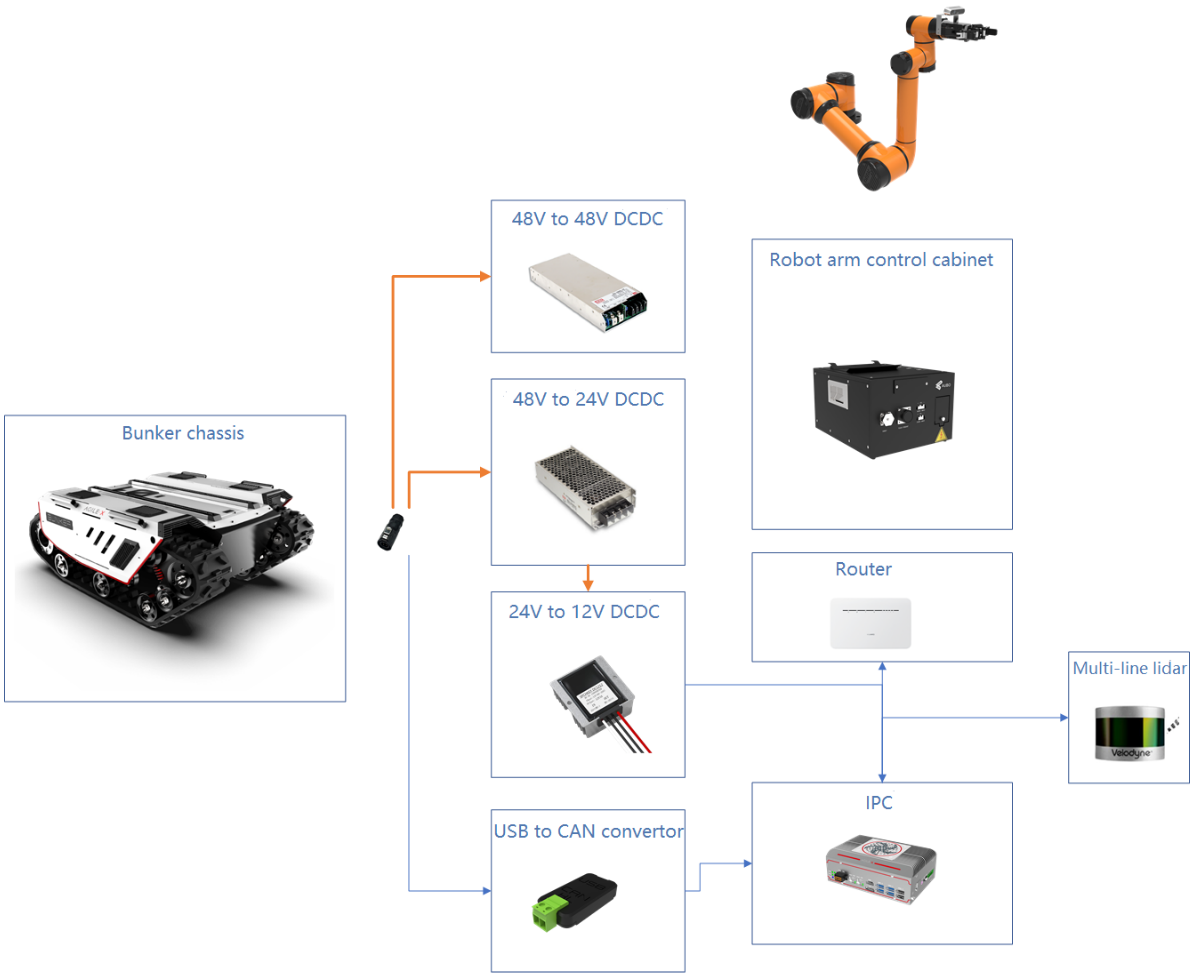
2 Hardware installation and electrical instructions

1. Hardware system composition

The tracked composite mobile robot project is composed of four parts as a whole, including the chassis of the tracked mobile robot, the control box, the robot arm and its sensing unit and interaction unit. The control cabinet mainly includes the IPC, router, voltage regulator module, mechanical arm teaching lamp composition. The specific location is shown in the figure below.



2.Hardware topology connection



3 Simulation（Coming soon）

4 Example of development

1.Pre-development preparation

1.Download the remote desktop tool

A remote desktop tool is installed in the IPC in the robot. Users need to install corresponding tools on their laptops or computers, and remotely control the IPC on the robot through the router on the robot.

（1）Downloading the Installation package

https://www.nomachine.com/download

Select your own laptop or the corresponding operating system to download.

（2）Usage

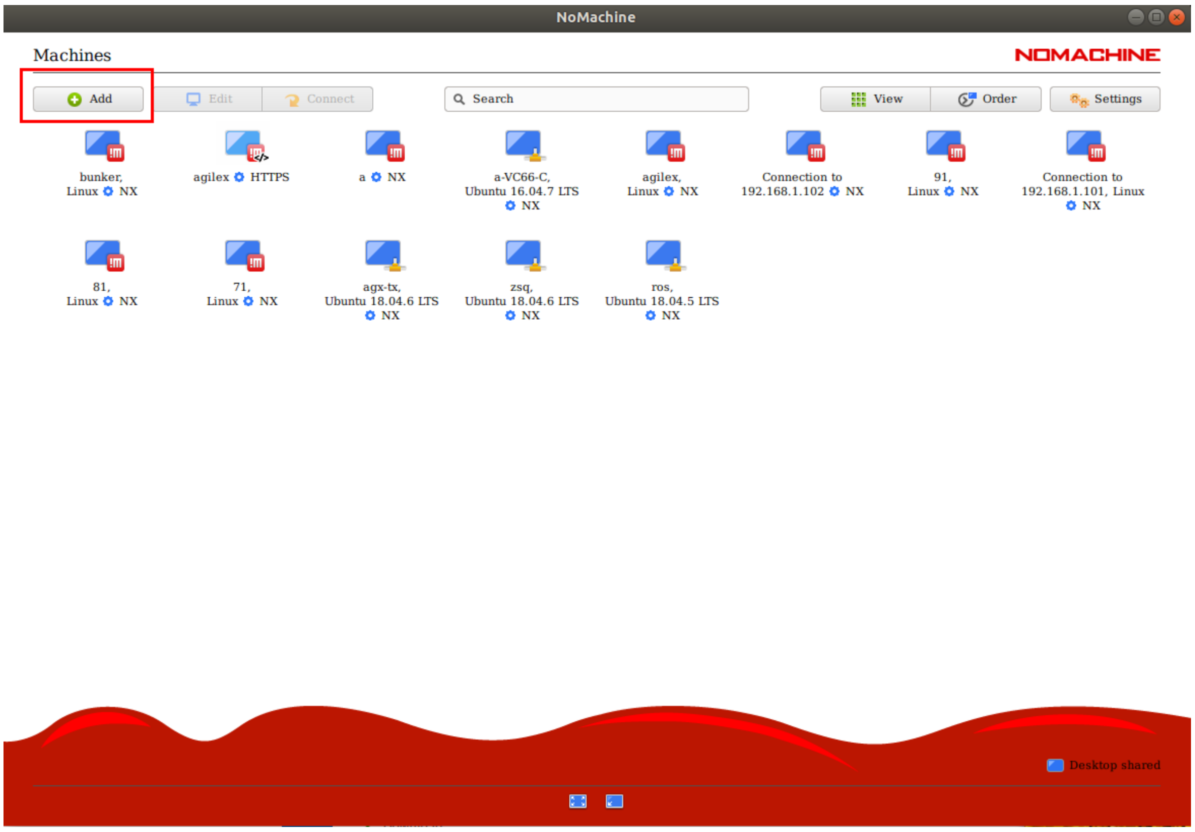
User name on IPC：bunker，Password：agx.

Router name：HAIWEI_B316_，Router's password and login password is the same：12345678，

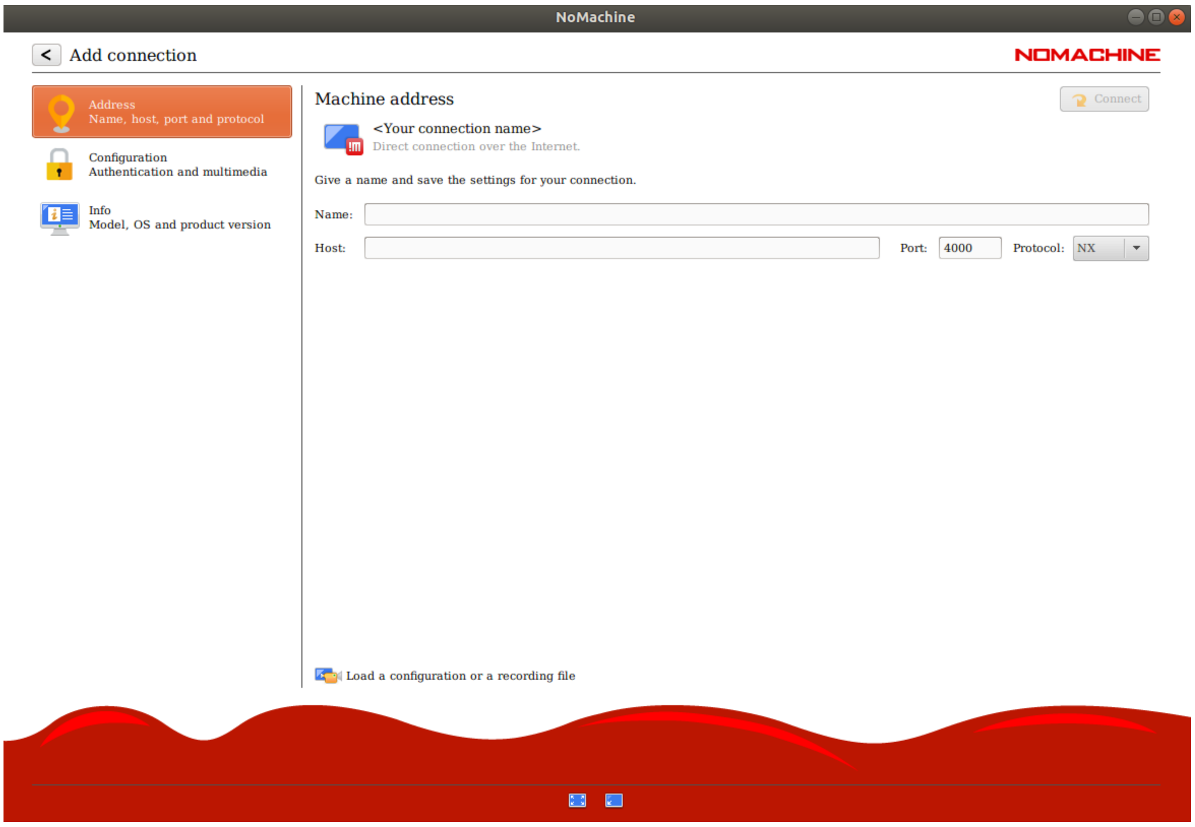
Router ip：192.168.1.1 ，IPC ip：192.168.1.101

First, use your computer or laptop to find the wifi on the robot and input your password to connect.

Open the downloaded remote connection tool Nomachine. Add a remote connection.


NoMachine
Machines
NOMACHINE
Add
Edit
Connect
Search
View
Order
Settings
bunker, Linux NX
agilex HTTPS
a NX
a-VC66-C, Ubuntu 16.04.7 LTS NX
agilex, Linux NX
Connection to 192.168.1.102 NX
91, Linux NX
Connection to 192.168.1.101, Linux NX
81, Linux NX
71, Linux NX
agx-tx, Ubuntu 18.04.6 LTS NX
zsq, Ubuntu 18.04.6 LTS NX
ros, Ubuntu 18.04.5 LTS NX
Desktop shared

name：bunker，host：192.168.1.102


NoMachine

Add connection

Address
Name, host, port and protocol

Configuration
Authentication and multimedia

Info
Model, OS and product version

Machine address

<Your connection name>
Direct connection over the Internet.

Give a name and save the settings for your connection.

Name:

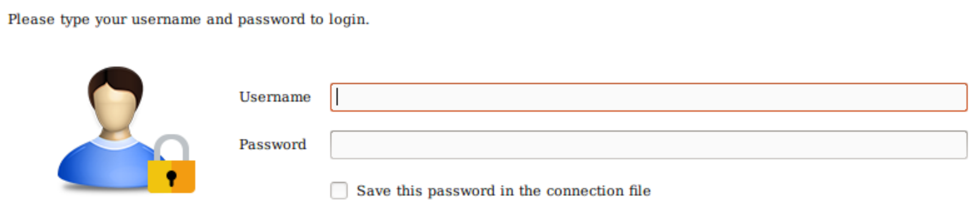
Host:

Port: 4000

Protocol: NX

Load a configuration or a recording file

Input username：bunker　　　password：agx，select to remember the password.


Please type your username and password to login.

Username

Password

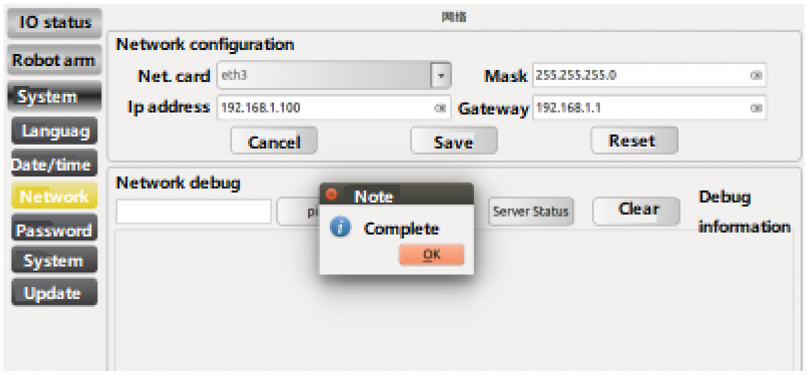
Save this password in the connection file

Then, you can connect it to the IPC on the robot.

2. IP setting

Set the network parameters of the robot arm：

- Enter【Setting】->【System】->【Network】；
- Choose the network card（Usually，there is only one option.）；
- Set robot IP in “IP address”，The robot IP must be on the same network segment as the IPC. That is the first three segments are the same with it. For example：192.168.1.100；
- In “Mask”，input：255.255.255.0；
- In “Gateway”，input：192.168.1.1 . The first three segments for Gateway will need to be the same as IP and the last segment is 1.
- Click 【Save】
- Restart the robot arm. This is very important！Because only the robot arm restarts，the new network parameters will be activated.



Network configuration

Net. card: eth3
Mask: 255.255.255.0
Ip address: 192.168.1.100
Gateway: 192.168.1.1

Complete

3. Use ar_track_alvar to achieve visual positioning and tracking

3.1 Function brief

This package is an open source AR tag tracking library Alvar's ROS PACKAGE . ar_track_alvar has 4 main functions：

（1）Generate AR tags with different sizes，resolutions，data/ID and encoding.

（2）Identify and track the pose of a single AR tag, optionally select integrating depth data from depth cameras for better pose estimation.

（3）Identify and track combination poses composed of multiple tags. This allows for more stable attitude estimation, robustness to occlusion, and tracking of multilateral objects.

（4）Automatically calculates the spatial relationships between tags in a bundle using camera images so that the user does not have to manually measure and enter tag locations in an XML file to use the bundle feature. (Currently not working.)

Alvar is newer and more advanced than ARToolkit, which has been the basis for several other ROS AR tag packages. Alvar has adaptive threshold processing to handle various lighting conditions, optical flow-based tracking for more stable attitude estimation, and an improved tag recognition method that does not slow down significantly as the number of tags increases.

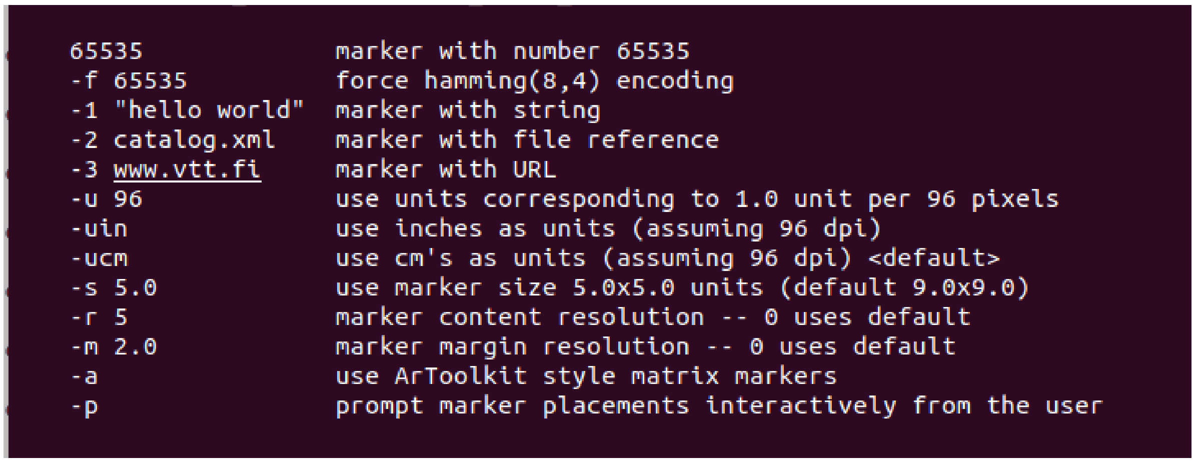
3.2 Instructions

(1) Generate tags

```
$ source ~/catkint_workspace/devel/setup.bash
$ rosrun ar_track_alvar createMarker 0 -s 3.0
```

> The following number indicates the ID number of the generated tags. You can generate tags with different numbers as required.

Detailed parameter Settings are shown as follows：



```
65535              marker with number 65535
-f 65535           force hamming(8,4) encoding
-1 "hello world"   marker with string
-2 catalog.xml     marker with file reference
-3 www.vtt.fi      marker with URL
-u 96              use units corresponding to 1.0 unit per 96 pixels
-uin               use inches as units (assuming 96 dpi)
-ucm               use cm's as units (assuming 96 dpi) <default>
-s 5.0             use marker size 5.0x5.0 units (default 9.0x9.0)
-r 5               marker content resolution -- 0 uses default
-m 2.0             marker margin resolution -- 0 uses default
-a                 use ArToolkit style matrix markers
-p                 prompt marker placements interactively from the user
```

（2）Print the tags

Note when printing tags, pay attention to the size. The default size is 3x3 cm. If the print size is different from the default size, please click:

~/your_workspace/src/open_manipulator_perceptions/open_manipulator_ar_markers/launch/agx_ar_pose. launch to modify the tags size in the file.

```
<arg name="user_marker_size"   default="3"/>
```

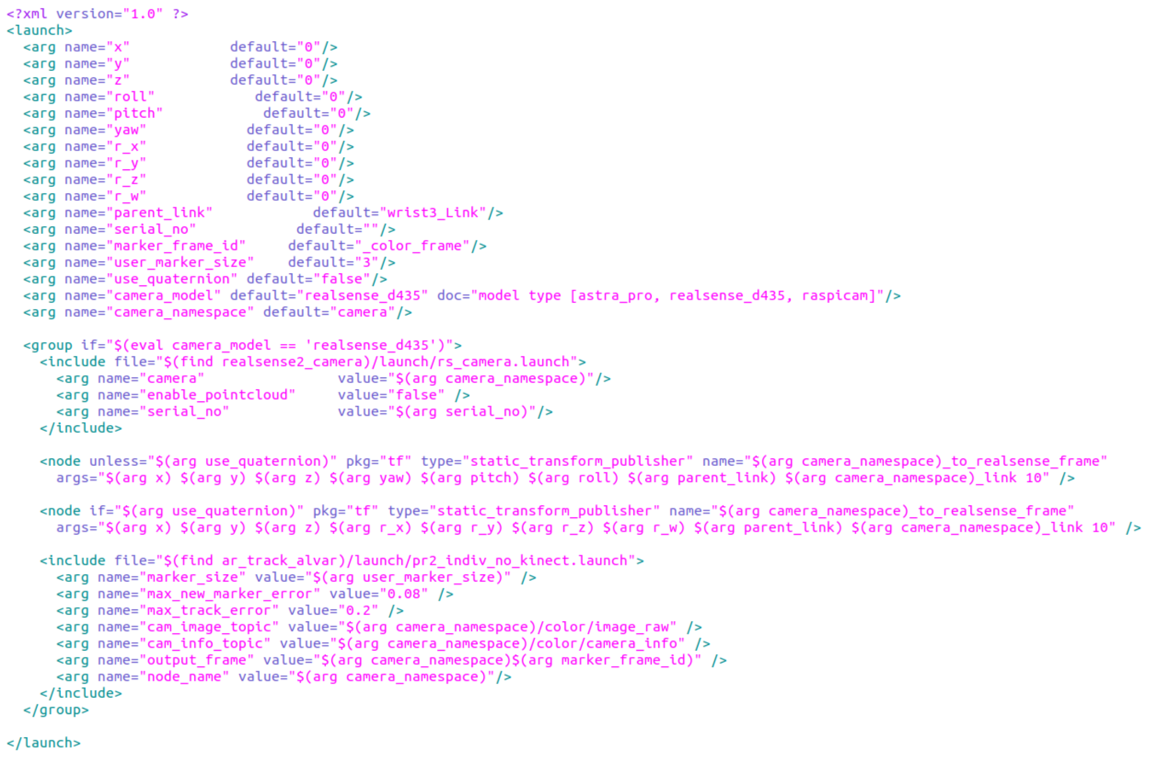
> Tag #0 is default on the object and tag #2 is on the object on the platform.

（3）Run the identification function

Use usb3.0 cable to connect the camera and computer，then run the below command：

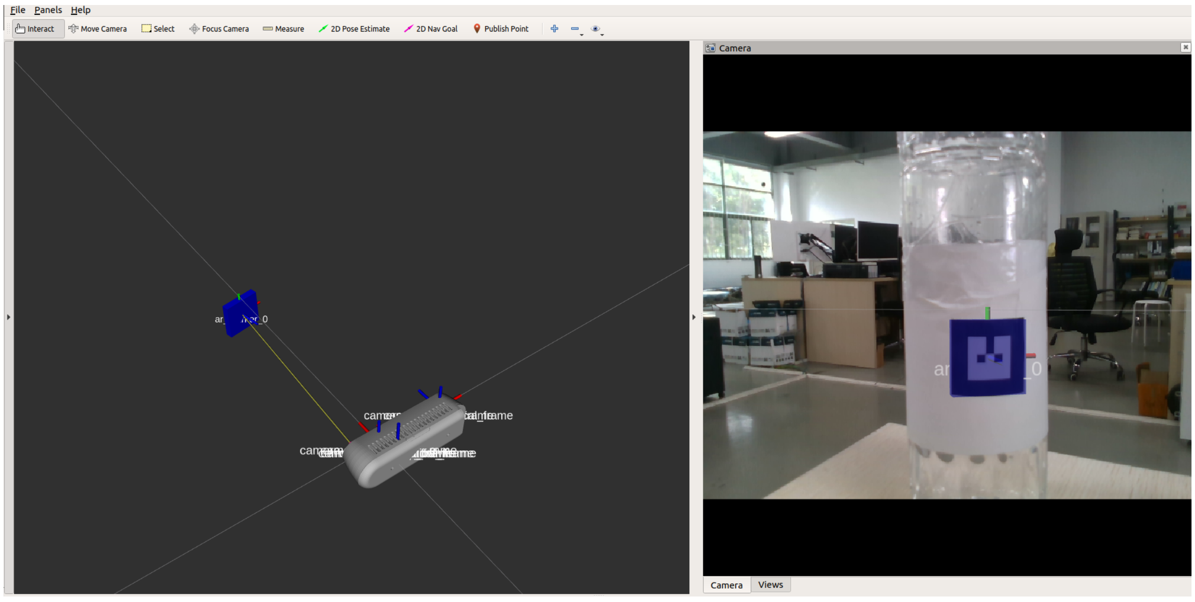
```
$source ~/catkin_workspace/devel/setup.bash
$roslaunch agx_aubo_bringup agx_ar_pose.launch
```

The file structure for agx_ar_pose.launch is shown in below figure and the detailed parameters are in the form below.



```
<?xml version="1.0" ?>
<launch>
  <arg name="x"            default="0"/>
  <arg name="y"            default="0"/>
  <arg name="z"            default="0"/>
  <arg name="roll"            default="0"/>
  <arg name="pitch"            default="0"/>
  <arg name="yaw"            default="0"/>
  <arg name="r_x"            default="0"/>
  <arg name="r_y"            default="0"/>
  <arg name="r_z"            default="0"/>
  <arg name="r_w"            default="0"/>
  <arg name="parent_link"            default="wrist3_Link"/>
  <arg name="serial_no"            default=""/>
  <arg name="marker_frame_id"     default="_color_frame"/>
  <arg name="user_marker_size"    default="3"/>
  <arg name="use_quaternion" default="false"/>
  <arg name="camera_model" default="realsense_d435" doc="model type [astra_pro, realsense_d435, raspicam]"/>
  <arg name="camera_namespace" default="camera"/>

  <group if="$(eval camera_model == 'realsense_d435')">
    <include file="$(find realsense2_camera)/launch/rs_camera.launch">
      <arg name="camera"                value="$(arg camera_namespace)"/>
      <arg name="enable_pointcloud"     value="false" />
      <arg name="serial_no"             value="$(arg serial_no)"/>
    </include>

    <node unless="$(arg use_quaternion)" pkg="tf" type="static_transform_publisher" name="$(arg camera_namespace)_to_realsense_frame"
      args="$(arg x) $(arg y) $(arg z) $(arg yaw) $(arg pitch) $(arg roll) $(arg parent_link) $(arg camera_namespace)_link 10" />

    <node if="$(arg use_quaternion)" pkg="tf" type="static_transform_publisher" name="$(arg camera_namespace)_to_realsense_frame"
      args="$(arg x) $(arg y) $(arg z) $(arg r_x) $(arg r_y) $(arg r_z) $(arg r_w) $(arg parent_link) $(arg camera_namespace)_link 10" />

    <include file="$(find ar_track_alvar)/launch/pr2_indiv_no_kinect.launch">
      <arg name="marker_size" value="$(arg user_marker_size)" />
      <arg name="max_new_marker_error" value="0.08" />
      <arg name="max_track_error" value="0.2" />
      <arg name="cam_image_topic" value="$(arg camera_namespace)/color/image_raw" />
      <arg name="cam_info_topic" value="$(arg camera_namespace)/color/camera_info" />
      <arg name="output_frame" value="$(arg camera_namespace)$(arg marker_frame_id)" />
      <arg name="node_name" value="$(arg camera_namespace)"/>
    </include>
  </group>

</launch>
```

Parameters	Default value	Function
x,y,z	0	The static position with camera fixed in the robot arm.
roll,pitch,yaw	0	Euler Angle representation of static attitude of camera fixed in robot arm.
r_w,r_x,r_y,r_z	0	The quaternion representation of the static attitude of the camera fixed in robot arm.
parent_link	wrist3_Link	The relative coordinate system of tf published by camera.
serial_no	Null	Serial number to activate the camera.
user_marker_size	3	Tag size（cm)
use_quaternion	false	Whether to use quaternion to represent attitude
camera_model	realsense_d435	Camera model
camera_namespace	camera	Camera namespace



4.Use GMapping to build a map

4.1 Instruction

The GMapping function package subscribs to the robot's depth information, IMU information and odometry information, and completes the configuration of some necessary parameters to create and output a probability-based two-dimensional grid map. The GMapping feature pack is based on the open source SLAM algorithm of the OpenSLAM community.

4.2 Run

```
$ roslaunch agilexpro open_lidar.launch
$ roslaunch agilexpro gmapping.launch
```

After the map is completed，save it in the directory：~/catkin_workspace/src/agilexpro/maps

```
$ roscd agilexpro/maps
$rosrun map_server map_saver -f map
```

> -f map：Here，map means the name our users built. The default loading map during navigation is the one named **map**. If our users use another names when they save the map，they will need to go to the directory：~catkin_workspace/src/agilexpro/launch/navigation_4wd.launch ，and then change the below map.yaml to their own map name.
>
> ```
> <node name="map_server" pkg="map_server" type="map_server" args="$(find
> agilexpro)/maps/map.yaml" output="screen">
> ```

5.Use move_base for navigation

5.1 Instruction

The move_base package provides an implementation of an action (see actionLib package) and then it will attempt to implement it by using the mobile base if the target was given in the world. The move_base node links global and local planners together to accomplish its global navigation task. The move_base node also maintains two cost maps, one for the global planner and the other for the local planner (see costmap_2d package), for navigation tasks.

5.2 Run

```
$roslaunch agilexpro open_lidar.launch
$roslaunch agilexpro navigation.launch
```

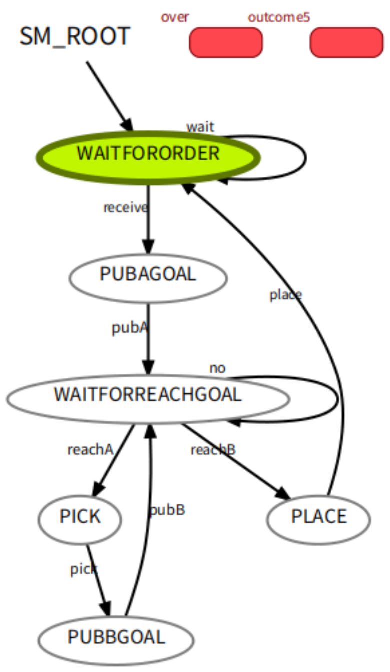
6. Use smach library to achieve tasks status switched.

6.1 Instruction

SMACH explicitly describes all possible states and state transitions, which is useful when a robot is performing some complex plans. This basically eliminates the conflict of putting different modules together, allowing systems such as mobile robot control to do more interesting things.

6.2功能运行

```
$source ~/catkin_workspace/devel/setup.bash
$ rosrun agx_aubo_smach smach_demo.py
```



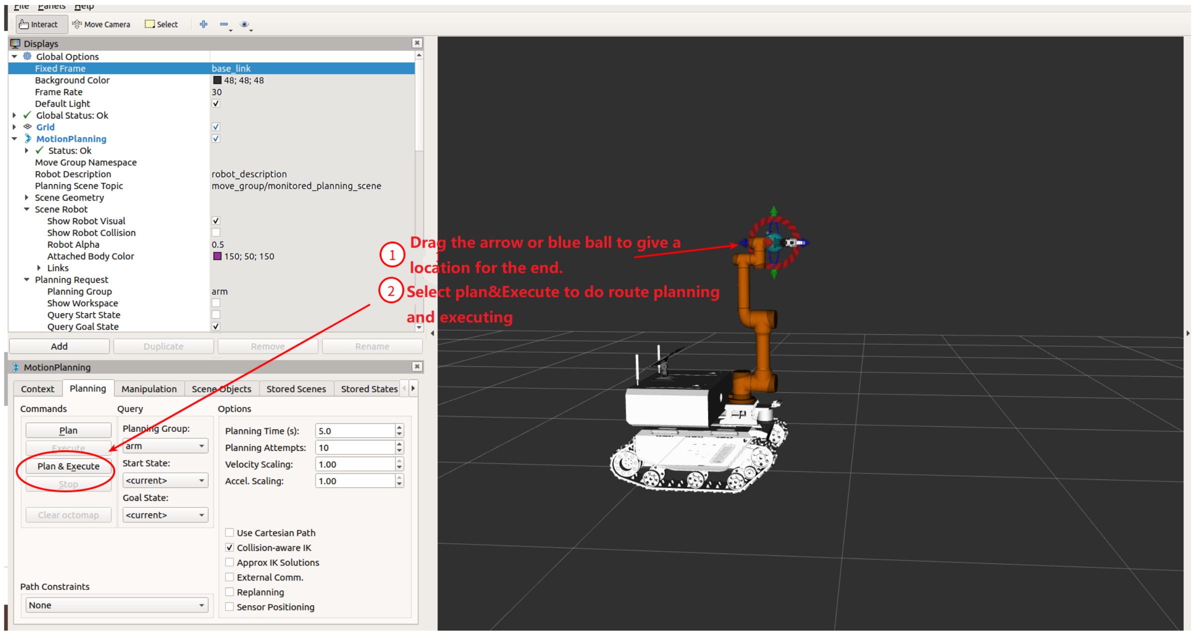
7.Use moveit for robot arm route planning

7.1 Instruction

Use moveit's motion planer to make the route planning，collision detection and route execution. Thus, when the target point is given, an optimal path can be calculated.

7.2 Run

```
$ roslaunch agx_aubo_bringup setup_arm.launch
```


① Drag the arrow or blue ball to give a location for the end.
② Select plan&Execute to do route planning and executing

Fixed Frame: base_link
Background Color: 48; 48; 48
Frame Rate: 30
Robot Description: robot_description
Planning Scene Topic: move_group/monitored_planning_scene
Robot Alpha: 0.5
Attached Body Color: 150; 50; 150
Planning Group: arm
Start State: <current>
Goal State: <current>
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 1.00
Accel. Scaling: 1.00
Collision-aware IK: ✓
Path Constraints: None

2. Run the function

1. Run robot arm

```
$ roslaunch agx_aubo_bringup setup_arm.launch
```

> Open the power supply of the robot arm control cabinet to unlock the safety of each joint of the arm. Long press the power button on the teaching display screen connected to the control cabinet of the robot arm and click Save -> Start in the small window that pops up to turn on the electrical switch of the robot arm.

In this launch file，we can change one parameter：robot_ip。The IP here is the default robot arm's control cabinet IP：192.168.1.100

```
<launch>
  <arg name="robot_ip" default="192.168.1.100" />
  <include file="$(find aubo_i5_moveit_config)/launch/moveit_planning_execution.launch">
    <arg name="robot_ip" value="$(arg robot_ip)"/>
  </include>
</launch>
```

2. Start camera

```
$ roslaunch agx_aubo_bringup open_camera.launch
```

> Note that the camera and IPC need to be connected with USB3.0 cable before startup.

The below parameters can be changed in launch file：

```
<arg name="camera_namespace" value="cam1"/>
  <arg name="serial_no" value="035422071503"/>
  <arg name="node_name" value="cam1"/>
<arg name="use_quaternion" value="true"/>
  <!--<arg name="roll" value="0"/>
  <arg name="pitch" value="-1.57"/>
```

```
    <arg name="yaw" value="-1.57"/>
    <arg name="x" value="0.05"/>
    <arg name="y" value="0.03"/>
    <arg name="z" value="0.0"/>-->
    <arg name="x" value="0.0446414171704"/>
    <arg name="y" value="0.053996778351"/>
    <arg name="z" value="0.0600140964856"/>
    <arg name="r_x" value="-0.000283027265062"/>
    <arg name="r_y" value="-0.0682970373998"/>
    <arg name="r_z" value="0.984025866108"/>
    <arg name="r_w" value="0.164403556559"/>
```

camera_namespace：Namespace of the camera node. It can be modified when starting multiple cameras.

serial_no：Camera serial number S/N code. When starting multiple cameras, we can use the SN to start the specify camera.

node_name：Generally the same as the namespace.

use_quaternion：true means it's using quaternions，false means it's using euler angle.

x,y,z：The position of the camera from the robot arm.

roll,pitch,yaw：Camera attitude euler Angle.

r_x,r_y,r_z,r_w：Camera attitude quaternion.

When launching the launch file, the image recognition node is also launched, which will update the spatial coordinate information of objects and object placement platform in real time. The two points of information can be obtained according to the following interface. The service name is /pick_point. The service type is agx_pick_msg/AgxPickSrv, and the message format is as follows:

```
int32 item     # 0--bottle 1--pick 2--place
int32 handpose  # The direction of claw grips the object 0--front 1--upside
int32 Prepose      # Prepare clip point. 0-- Real position of object;
             # 1--Forward or Shift upwards a position of the gripper according to the handpose
---
#Return
geometry_msgs/Point position
 float64 x
 float64 y
 float64 z
geometry_msgs/Quaternion orientation
 float64 x
 float64 y
 float64 z
 float64 w
```

其中q为0时，返回物体的坐标信息；q为2时，返回物体摆放位置信息。

3. Start robot arm claw

```
$ roslaunch agx_aubo_bringup open_gripper.launch
```

The topic interface to control the claw is /gripper/ctrl, message type is dh_gripper_msgs/GripperCtrl，Message format is as follow：

```
bool initialize       # Whether init
float32 position     # Distance between the fingers
float32 force         # Force to grip the object
float32 speed        # This parameter is invalid for two fingers claw.The clamping speed is positively
correlated with force
```

In the Agx_aubo_pick feature pack, there is a demo file that can input the finger position to control the claw. The instructions are as follows:

```
$ rosrun agx_aubo_pick control_gripper
```

4. Start agx_aubo_pick node

```
$ roslaunch agx_aubo_bringup agx_aubo_bring.launch
```

The node needs to acquire the taskid from the state machine to control the robot to perform different tasks. The service interface of the task is /send_task and the service type is agx_pick_msg/TaskCmd. The message format is as follows:

```
int32 taskID
geometry_msgs/PoseStamped A_goal
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
geometry_msgs/PoseStamped B_goal
  std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
---
bool result
```

5. Start smach states machine node

```
$ rosrun agx_aubo_smach smach_demo.py
```

Switch of states are defined below:

```
smach.StateMachine.add('WAITFORORDER', WaitFororder(),
     transitions={'receive':'PUBAGOAL',
  'wait':'WAITFORORDER'})

smach.StateMachine.add('PUBAGOAL', PubAGoal(),
     transitions={'pubA':'WAITFORREACHGOAL'})

smach.StateMachine.add('WAITFORREACHGOAL', WaitForReachGoal(),
   transitions={'no':'WAITFORREACHGOAL',
'reachA':'PICK',
'reachB':'PLACE'})

smach.StateMachine.add('PICK', Pick(),
     transitions={'pick':'PUBBGOAL'})

smach.StateMachine.add('PUBBGOAL', PubBGoal(),
     transitions={'pubB':'WAITFORREACHGOAL'})

smach.StateMachine.add('PLACE', Place(),
     transitions={'place':'WAITFORORDER'})
```

The above is the whole system state transition logic, you can add the corresponding state according to the need. Take the first state, where a "WAITFORORDER" state is defined with a WaitFororder class, and transitions represent jump of the state. If the output of WaitFororder is receive, it jumps to "PUBAGOAL", and if the output of WaitFororder is wait, it jumps to "WaitFororder".

6. Start mapping

```
$ roslaunch agilexpro open_lidar.launch
$ roslaunch agilexpro gmapping.launch
```

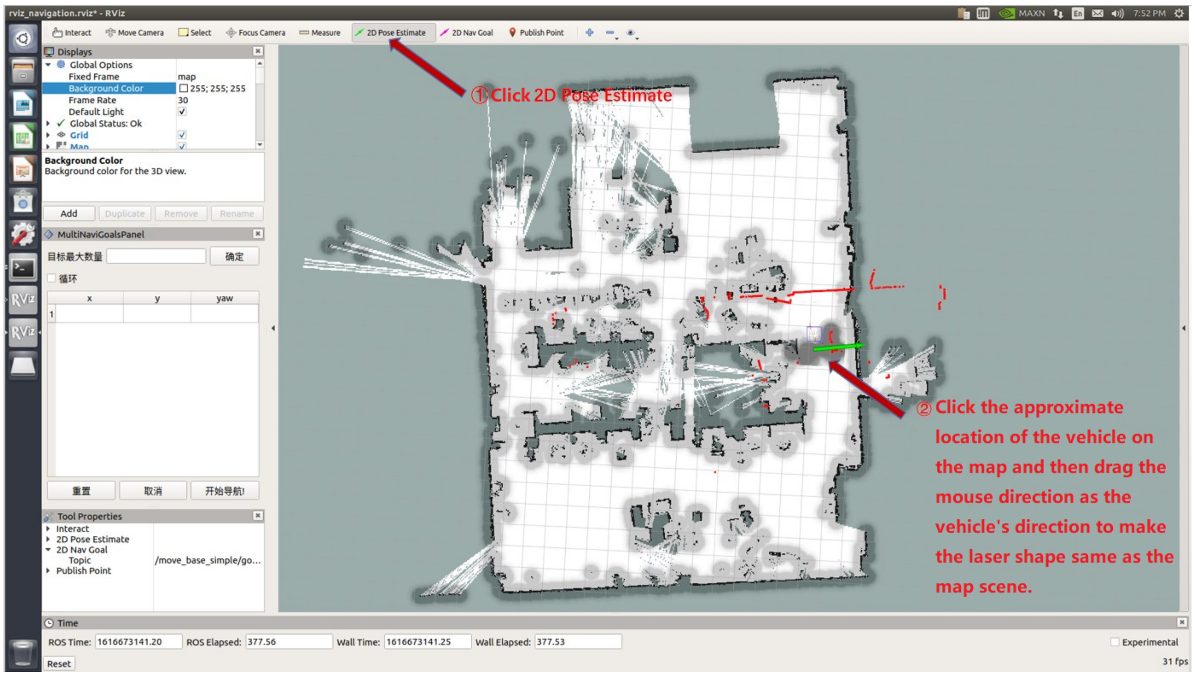
Use the remote control robot to build a map in the scene, and execute the following instructions to save the map.

```
$ roscd agilexpro/maps
$rosrun map_server map_saver -f map
```

7. Start navigation

```
$ roslaunch agilexpro open_lidar.launch
$ roslaunch agilexpro navigation_4wd.launch
```

When first start up, the program will default to the location where the mapping was started . You will need to publish an approximate location and use remote controller to rotate the chassis to calibrate. When the laser shape overlaps with the scene shape in the map, the correction is complete.


rviz_navigation.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
①Click 2D Pose Estimate
②Click the approximate location of the vehicle on the map and then drag the mouse direction as the vehicle's direction to make the laser shape same as the map scene.

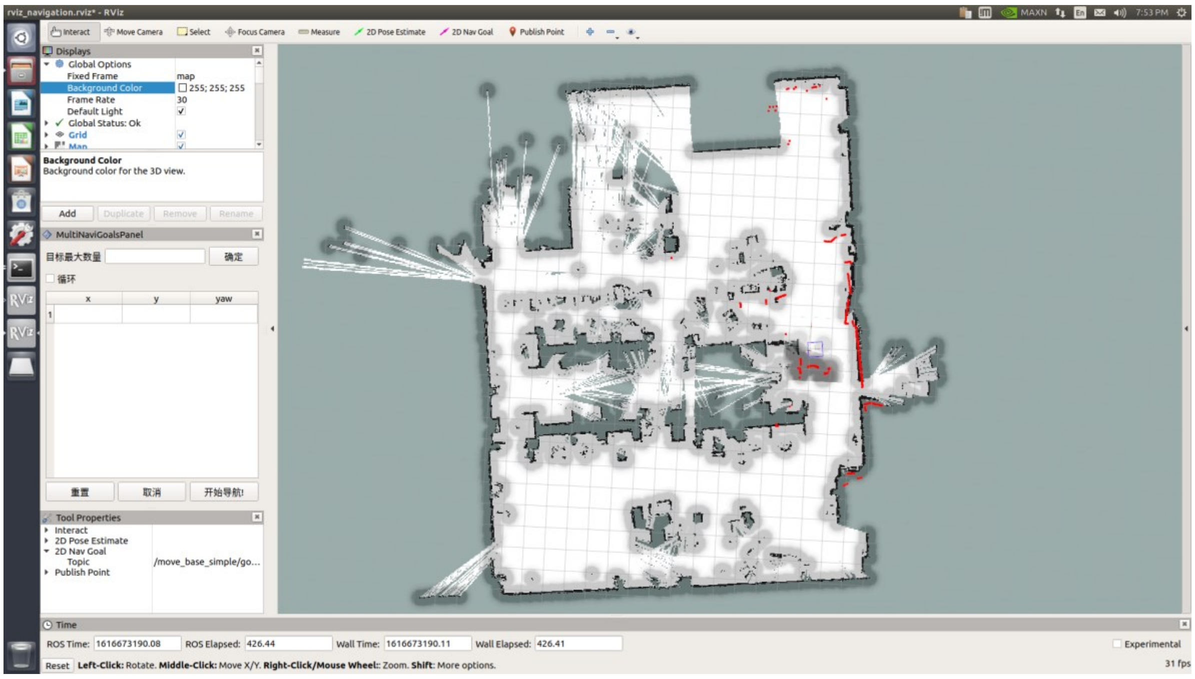
Displays
Global Options
Fixed Frame: map
Background Color: 255; 255; 255
Frame Rate: 30
Default Light: ✓
Global Status: Ok
Grid ✓
Map ✓

Background Color
Background color for the 3D view.

MultiNaviGoalsPanel
目标最大数量 确定
循环

Tool Properties
Interact
2D Pose Estimate
2D Nav Goal
Topic: /move_base_simple/go...
Publish Point

ROS Time: 1616673141.20 ROS Elapsed: 377.56 Wall Time: 1616673141.25 Wall Elapsed: 377.53



Displays
Global Options
Fixed Frame: map
Background Color: 255; 255; 255
Frame Rate: 30
Default Light: ✓
Global Status: Ok
Grid ✓
Map ✓

Background Color
Background color for the 3D view.

MultiNaviGoalsPanel
目标最大数量 确定
循环

Tool Properties
Interact
2D Pose Estimate
2D Nav Goal
Topic: /move_base_simple/go...
Publish Point

ROS Time: 1616673190.08 ROS Elapsed: 426.44 Wall Time: 1616673190.11 Wall Elapsed: 426.41

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

The topic interface for navigation is：/move_base_simple/goal，Message type is：geometry_msgs/PoseStamped，Message format is as follow：

```
std_msgs/Header header
 uint32 seq
 time stamp
 string frame_id
geometry_msgs/Pose pose
 geometry_msgs/Point position
  float64 x
  float64 y
  float64 z
 geometry_msgs/Quaternion orientation
  float64 x
```

```
    float64 y
    float64 z
    float64 w
```

Feedback information can be obtained from the topic **/move_base/result topic** when it arrives to a point. The message type is： base_msgs/MoveBaseActionResult . The message format is as follow：

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
actionlib_msgs/GoalStatus status
  uint8 PENDING=0
  uint8 ACTIVE=1
  uint8 PREEMPTED=2
  uint8 SUCCEEDED=3
  uint8 ABORTED=4
  uint8 REJECTED=5
  uint8 PREEMPTING=6
  uint8 RECALLING=7
  uint8 RECALLED=8
  uint8 LOST=9
  actionlib_msgs/GoalID goal_id
    time stamp
    string id
  uint8 status
  string text
move_base_msgs/MoveBaseResult result
```

When the robot arrives at the target point, the status is 3.

8. Record the coordinates of two points

```
$ roscd agx_aubo_pick/config
$ rosrun agx_aubo_pick record
```

In coordination with Step 7, move the trolley control to the specified position, such as in front of the workpiece, about 1m away. As prompted, press Enter to record the location of the two points.

9. Mobile grab demonstration

Set two locations as a clip point and placing point in the scene. After recording the coordinate positions of the two locations through Step 8, enable the following command：

```
$ roslaunch agx_aubo_bringup setup_arm.launch              #Start robot arm
$ roslaunch agx_aubo_bringup open_camera.launch            #Start camera
$ roslaunch agx_aubo_bringup open_gripper.launch           #Start claw
$ roslaunch agx_aubo_bringup agx_aubo_bring.launch         #Start moving claw node
$ rosrun agx_aubo_smach smach_demo.py                      #Start states machine
$ roslaunch agilexpro open_lidar.launch                    #Start LIDAR
$ roslaunch agilexpro navigation_4wd.launch                #Start navigation
$ roslaunch agx_aubo_pick  test.launch                     #Start execution
```

When the remote controller is switched to command control mode, the Cobot starts to navigate to the first location to grab things.

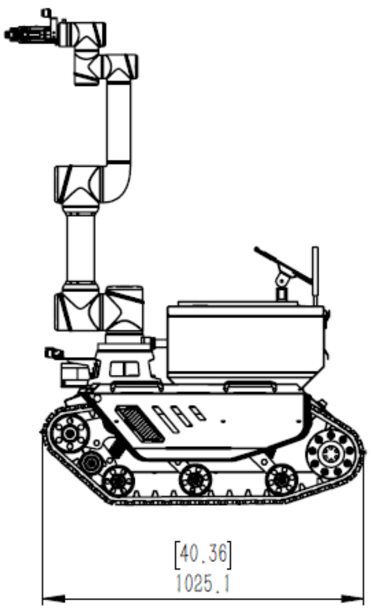
10. In place grab and place demonstration

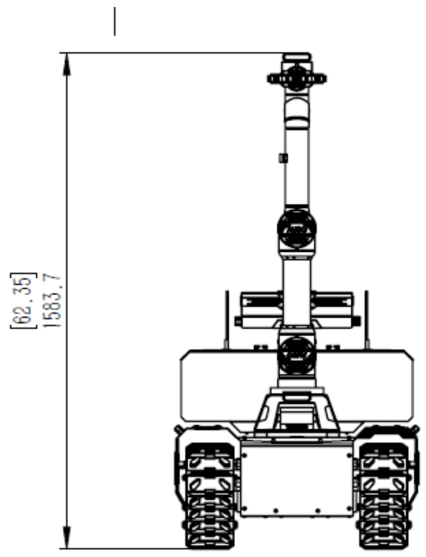
Label the grasping object and place it in a suitable position of the robot. Execute the following commands to grab and place the demo：

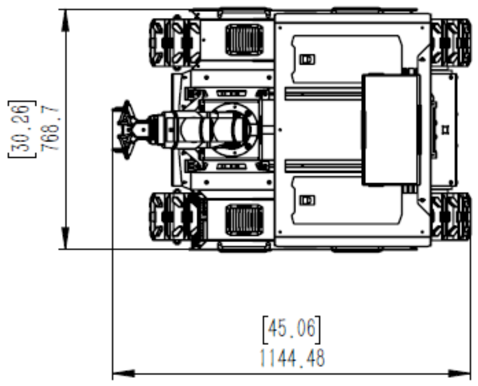
```
$ roslaunch agx_aubo_bringup set_setup_arm.launch
$ roslaunch agx_aubo_bringup open_camera.launch
$ roslaunch agx_aubo_bringup open_gripper.launch
$ rosrun agx_aubo_pick demo_pick    #Grab objects
$ rosrun agx_aubo_pick demo_place     #Place objects
```

3 FAQ and troubleshooting

4 Figures and figure legends







5 Others

1. [User manual and supporting instruction manual of Aubo robot arm]:
2. VLP16 LIDAR user manual.