AgileX Robotics Autoware Open Sourse Autonomous Kit

1. Basic hardware configuration list

a) Computing unit and accessories

No.	Accessories	Model	Quantity
1	Computing unit	ASUS VC66 (i7-9700 16G 512G M.2 NVME + Solid	1
2	Computing power adapter	24v to 19V (10a)	1
3	A set of mouse and keyboard		1
4	14-inch wireless screen		

b) Perception equipment and accessories

No.	Accessories	Model	Quantity
1	Multi line lidar	Robosense RS16	1
2	24V VRM	24v to 19V (10a)	

c) Integrated navigation and accessories (optional)

No.	Accessories	Model	Quantity
1	Integrated navigation	Newton M2	1
2	RF connecter		2
3	Data/Power connecter		1
4	GPS aerial		2
5	Base of the aerial		2

d) Chassis platform

No.	Accessories	Model	Quantity
1	Chassis mobile	HUNTER/HUNTER 2	1

	platform		
2	Remote Controller	FS i6s	1
3	USB to CAN	CAN analyzer	1

Note: Due to the different network standard in different countries and regions, unfortunately the router is not able to provide within the accessories list, users can purchase the router based on the requirement by themselves.

2. Basic function of Autoware development kit and description of ODD

1) Basic function of development kit

- Introduction to the wire control of the chassis
- Control the chassis by ROS
- View the lidar 3D point cloud data based on Robosense RS16
- Use Autoware to build 3D point cloud map, and view 3D point cloud data
- Use Autoware to record path points
- Use Autoware to follow path points
- Use Autoware to follow path points (obstacle avoidance)
- Use hybrid A* for free navigation (static detour)
- Use Autoware for local part path planning (set up multiple lane changes)
- Edit vector maps (lane lines, zebra crossings, curbs, etc.)
- Use Autoware for global path planning (combined with vector map)

2) Description of ODD

Item	**Content**
Application conditions	Indoor and outdoor environment
Applicable road situation	Clear Roads and limited situation(security, logistics, autonomous driving)
Weather	Regular weather such as sunny, cloudy, and foggy (visibility above 100 meters)
Pavement requirements	1. Smooth and relative clean roads (asphalt roads, cement roads, etc.), excluding construction roads with many protrusions or depressions or roads with many scattered objects; 2. Wet environment (the depth of water should not exceed the height of the chassis bottom for 5cm, not putting in the water); 3. Less than 10° slope is recommended when climbing up (it can be increased appropriately according to the different chassis drive capacity);

Valid period	The period which there is enough sunshine during the day and high visibility at night
Speed	≤10km/h
Working humidity	0~80%

3. Basic Introduction:

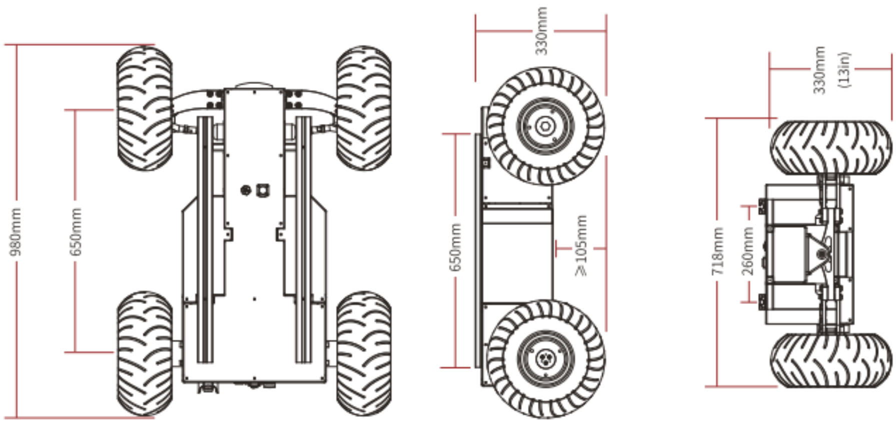
i. Hardware Introduction

1) Chassis platform

HUNTER1.0

HUNTER1.0 is a programmable UGV with Ackermann steering model which its chassis is based on Ackermann steering theory. Therefore, it is similar to normal cars and high performances on cement and asphalt roads . Compared with the four-wheel differential chassis, HUNTER chassis has higher performances for load carrying and speed, also it causes less abrasion for the structure and tires. Although HUNTER is not designed for all kinds of terrains, it is equipped with a swing arm suspension which is able to go through some normal obstacles such as speed bumps, etc. Additional extension such as stereo camera, laser lidar, GPS, IMU and robotic manipulator can be installed on HUNTER optionally. HUNTER is mostly used for autonomous driving education ,research, indoor and outdoor security patrolling, environment sensing, general logistics and transportation.

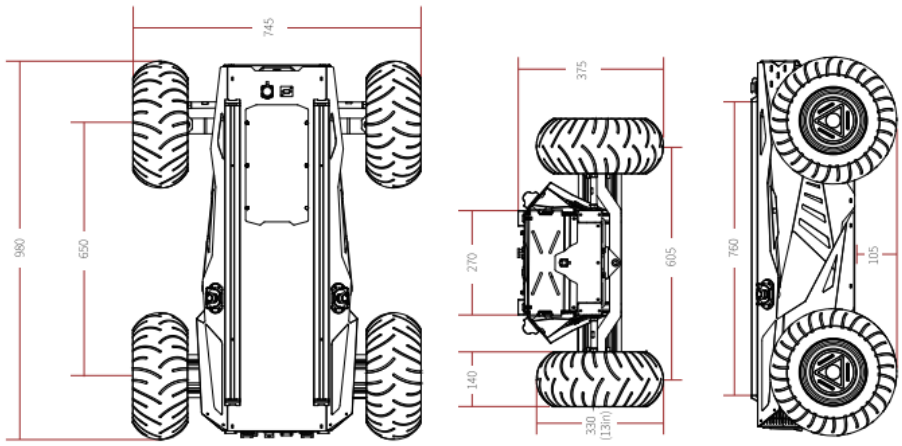
Parameter Types	Items	Values
Mechanical specifications	L × W × H (mm)	980 X 718 X 330
	Wheelbase (mm)	650
	Front/rear wheel base (mm)	578
	Weight of vehicle body (kg)	45~50
	Battery type	Lithium battery 24V 20aH
	Power drive motor	DC brushless 2 X 200W
	Steering drive motor	DC brushless 200W
	Reduction gearbox	1:30
	Drive type	Rear wheel drive
	Steering	Front wheel Ackermann
	Maximum steering angle	30°
	Steering accuracy	0.5°
Motion	No-load highest speed (m/s)	1.5
	Minimum turning radius (mm)	1700
	Maximum climbing capacity	20°
	Minimum ground clearance (mm)	105
Control	Control mode	Remote control Control command mode
	RC transmitter	2.4G/extreme distance 1km
	Communication interface	CAN / RS232



HUNTER 2.0

HUNTER2.0 was born for low-speed self-driving which based on front-wheel Ackerman steering theory and swing arm suspension, is able to pass different kind of obstacles, secondary development interfaces and standard installation components are making HUNTER2.0 the best solution for mobile robot self-driving program. Compared with HUNTER1.0, the upgraded version HUNTER2.0 has gradient parking function which achieved long-term ramp parking. If the vehicle is powered off or malfunctions while driving on a sloped road, the tires will be locked, making it stable and reliable. HUNTER2.0 has lithium iron phosphate battery and the capacities can be customized based on the requirement. The speed also can be customized up to 10km/h, meet the requirements of different autonomous driving scenarios.

Parameter Types	Items	Values
Mechanical specifications	L × W × H (mm)	980 × 745 × 380
	Wheelbase (mm)	650
	Front/rear wheel base (mm)	605
	Weight of chassis body (kg)	65/70
	Battery	Lithium battery 24V 30Ah/60Ah
	Power drive motor	DC brushless 2 ×400W
	Steering drive motor	DC brushless 200W
	Reduction gearbox	1:40
	Drive system form	Power off electromagnetic band type brake
	Steering	Front wheel Ackermann
	Maximum steering angle	33°
	Steering accuracy	0.5°
Motion	No- load MAX speed (m/s)	1.5
	Minimum turning radius (mm)	1.6
	Maximum climbing capacity	10°
	Minimum ground clearance (mm)	105 (Angle 30°)
Control	Control mode	Remote control Control command mode
	RC transmitter	2.4G/extreme distance 1km
	System interface	CAN



2) Robosense Introduction

RS-LiDAR-16 uses 16 laser heads to simultaneously emit high-frequency laser beams to continuously scan the external environment. Because it has high-speed digital signal processing technology and ranging algorithms to acquire three-dimensional space point cloud data and object reflectivity rate, so that the machine is able to observe the surrounding and highly capable for location navigation and obstacles avoidance.

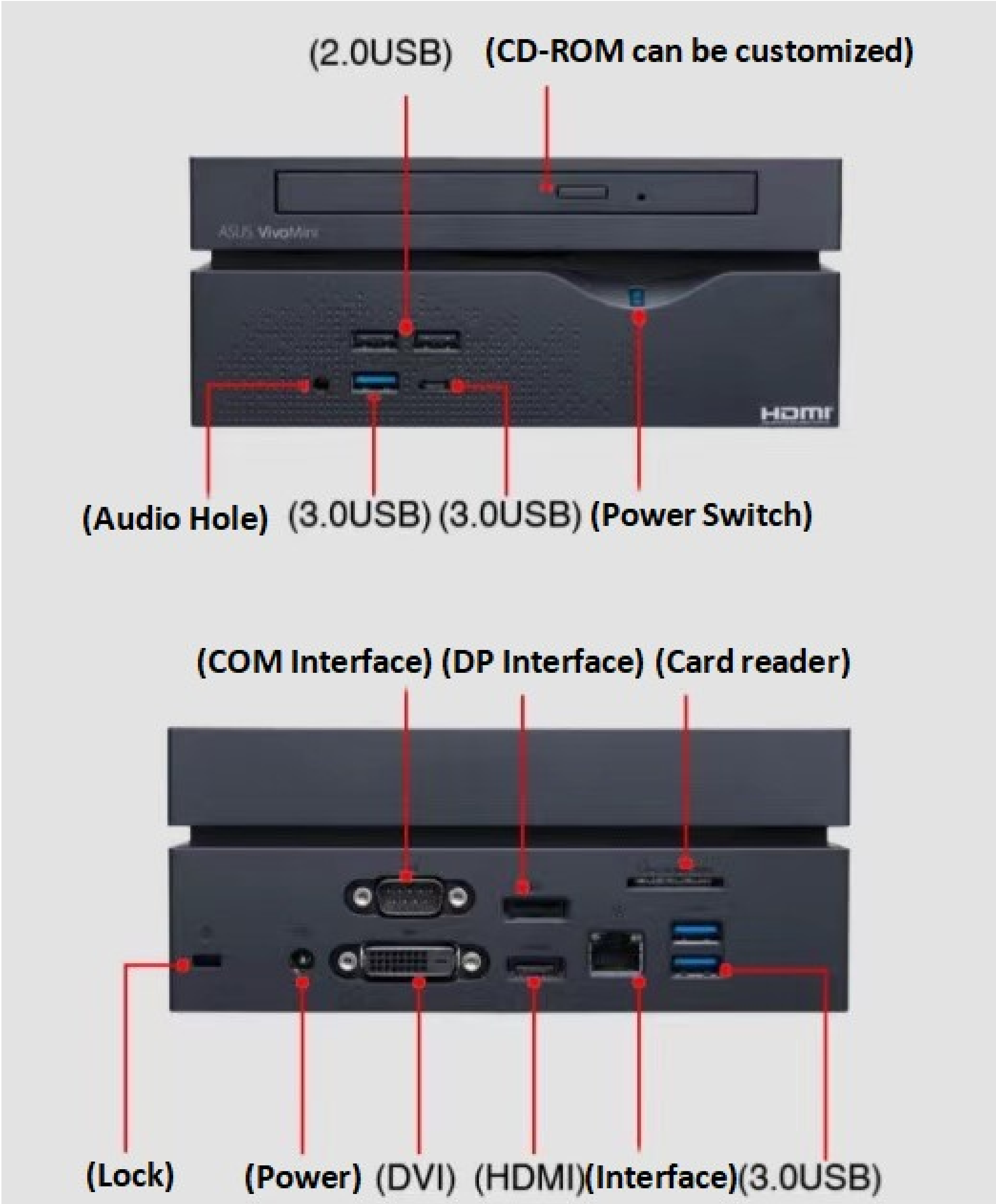
Figure 2 Robosense lidar

Sensor	● TOF method ranging 16 channel ● Range: 20cm-150m (Target reflectivity rate 20%)

	● Precision:+/-2cm (Typical value) ● Visual angle (Vertical): ±15° (Total 30°) ● Angular resolution: (Vertical): 2° ● Visual angle (Horizontal) : 360° ● Angular resolution (Horizontal/azimuth): 0.09° (5Hz) to 0.36° (20Hz) ● Speed: 300/600/1200rpm (5/10/20Hz)
Laser	● Class 1 ● Wavelength: 905nm ● Laser launch angle: Horizontal 3mrad, Vertical 1.2mrad
Output	● 320kBytes/s ● 100M Internet ● UDP include Distance information Rotation angle information Calibrated reflectivity information Synchronized time label (Resolution ratio 1us)
Mechanical/electronic operation	● Power consumption: 9w (Typical value) ● Operational voltage: 12VDC (With interface box, stable voltage) ● Size: Diameter 109mm * Height 82.7mm ● Protective safety level: IP67 ● Operational temperature range: -10°C~+60°C

3) Introduction of computing unit

The computing unit uses intel i7-9700 processor which main frequency is eight-core and eight-wire 3Ghz , up to 32 GB memory and two hard disks.

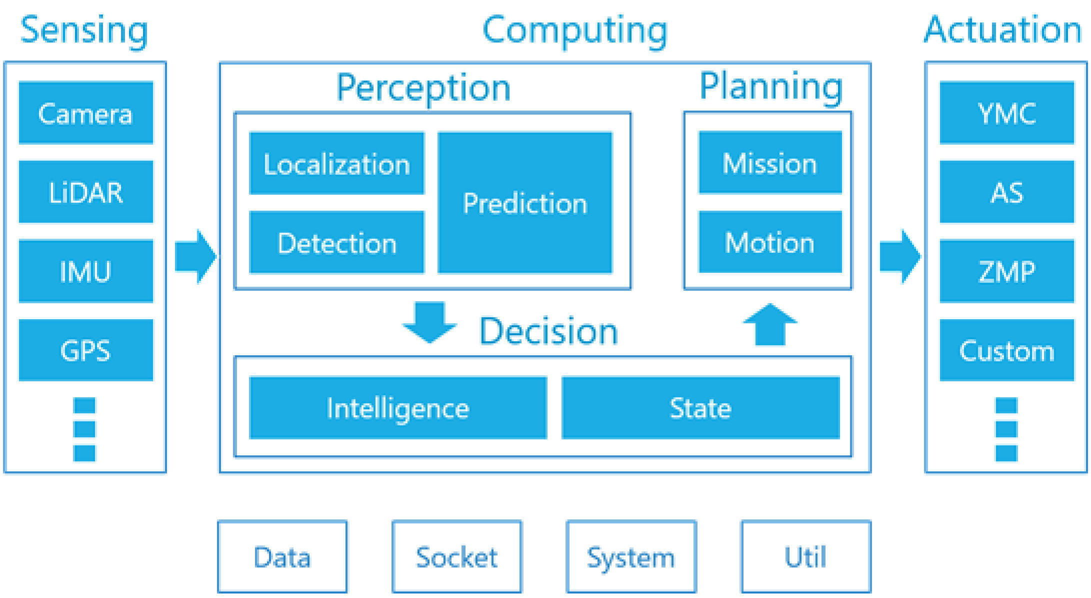


ii. System Architecture

1) Introduction of Autoware system

Autoware is the first open source integration software for autonomous driving vehicle in the world. Autoware is mainly suitable for cities, but also applicable to highway and non-municipal roads . At the same time, there are development and application resources on the Autoware open source software which is built on ROS operating system. The first official version was released by the Nagoya University research team with the leadership of Prof. Shinpei Kato in August 2015. In late December 2015, in order to maintain Autoware and apply it to real self-driving cars, Prof. Shinpei Kato founded Tier IV . As the time goes on, Autoware has became an open source project acknowledged by the public. Autoware is also the first "all in one" open source software for autonomous driving technology in the world.

Autoware contains the required function modules. In this manual, there are only general concept for function modules, customers are welcome to develop detail research by their own.



Perception

Autoware support camera, LiDAR, IMU and GPS as the main sensor. From the technical view, if it is not verified, as long as the sensor driver software is provided, almost all kinds of cameras, LiDAR, IMU and GPS can be applicable in Autoware.

Computing/Perception

The perception ability of Autoware is consisted of localization, perception and prediction. Though combining 3D maps with SLAM algorithms of GNSS and IMU sensor to achieve localization. Perception contains sensor fusion algorithm and deep neural networks camera and Lidar. Prediction is based on the results of localization and perception.

Localization

lidar_localizer use scan data from LiDAR and pre-install 3D map information to calculate self-driving car position in global coordinate (x, y, z, roll, pitch, yaw). We suggest to use the NDT algorithm to match the LiDAR scan with the 3D map, and the ICP algorithm is also applicable.

gnss_localizer converts the NMEA messages from GNSS receiver to the (x, y, z, roll, pitch, yaw) position. This result can be used as the location of the autonomous vehicle independently, or it can be used to initialize and supplement of the lidar_localizer result.

Generally, dead_reckoner uses IMU sensors to predict the next frame position of the autonomous vehicle, and interpolates the results of lidar_localizer and gnss_localizer.

Perception

Lidar_detector acquires the point cloud data from 3D laser scanner and it has object detecting function which based on LiDAR. The Euclidean clustering algorithm supports the basic performance which is able to find the clusters of LiDAR scans (point clouds) above the ground. In order to classify clusters, it support the algorithm base on DNN such as VoxelNet and LMNet.

vision_detector acquires image data from the camera and it has object detection function which is based on image. Main algorithm includes R-CNN, SSD and Yolo which are designed to single DNN executing to achieve actual-time performance. and support various different detection types, such as cars and passengers.

vision_tracker is able to track the results of vision_detector. This algorithm is based on Beyond Pixels. Project the tracking result on the image platform, and combine it with the result of lidar_detector in 3D space by using Fusion_tools.

fusion_detector requires the point cloud data from laser scanner and image data from camera, and achieve accurate target detection in 3D space. The position of laser scanner and camera must be calibrated in advance. The current implementation is based on the MV3D algorithm, this network has less extensibility compared with the original algorithm.

fusion_tools are able to combine the result of lidar_detector and vision_tracker. The information identified by vision_detector is add to the point cloud cluster detected by lidar_detector.

object_tracker is the motion of the object detected and recognized by the above procedure. The tracking results can be used for object behavior prediction in the future and object velocity evaluation. The tracking algorithm is based on a Kalman filter. Another variant also supports particle filters.

Prediction

object_predictor uses the results of the above object tracking to predict the future route of moving objects (such as cars and passengers).

collision_predictor use the results of object_predictor to predict whether the autonomous car is going to collide with any kinds of object in motion. In addition to the results of object tracking, the information of route trajectory and speed of the autonomous vehicle is also required as input data.

cutin_predictor use the same information as collision_predictor did to predict whether there is any neighbour car cut in front of the autonomous vehicle.x

Computing/Decision

Autoware's decision-making module contains perception and planning modules. According to the result of perception, the driving behavior of Autoware is represented by the finite state machine, so that the appropriate planning function can be selected. The current decision-making method is based on the rule system.

Computing/Planning

The last module of Autoware is the planning module which function is making plans for global tasks and local (at the time) movement based on the results of the perception and decision-making modules. Generally, the global task is determined when the autonomous vehicle is started or restarted, and the local motion is updated based on the state changing. For example, if the state of Autoware is set to "Stop", the plan is setting the speed of the autonomous vehicle to zero in front of an object with a safety margin or stop at the stop line. Another example is that if the state of the automatic software is set to "Avoid", the trajectory of the autonomous vehicle is planned to pass the obstacle. The main software packages included the planning module as follows.

Planning

·route_planner searches for the global route to the destination. The route is represented by a set of intersections in the network.

·lane_planner determines to use which lanes and the route generated by route_planner. The lane is represented by a set of road signs and multiple road signs (each road sign is corresponding to a lane) generated by this package.

·waypoint_planner can be used to generate a set of guide points to the destination. The difference between this package and lane_planner is that it generates a single way point instead of an array of way points.

·waypoint_maker is a practical tool for saving and loading manual way points. If it is needed to save way points to a specific file, please drive the vehicle manually after activating localization, Autoware will record the way points and speed information of the driving route. You can download the recorded way points from the specific file later, so that the motion planning module is able to follow the path.

Motion

·velocity_planner get updates from lane_planner, waypoints_planner or waypoints_maker

Speed plans for way points is slow/accelerate vehicles for different road circumstances, such as stop lines and traffic lights. Please note that the speed information embedded in a given waypoint is static, and the package will update the speed plan based on the driving circumstances.

astar_planner executes the hybrid A* search algorithm, this algorithm generates the path from current position to specific position. The software package can be used to avoid obstacles and make sudden turns on given way points as well as route selection in free spaces such as parking lots.

adas_lattice_planner execute the state lattices planning algorithm. The algorithm is based on a spline curve, a predefined parameter table and ADAS mapping (also known as vector mapping) information generates multiple feasible trajectories before the current position. The software package is used for obstacle avoidance and lane changing.

waypoint_follower executes the Pure Pursuit algorithm. The algorithm generates a set of twisted velocities and angular velocities (or positive angles) to move the autonomous vehicle to a target waypoint on a given waypoint in circular motion. This package should be used in combination with velocity_planner, astar_planner and/or adas_lattice_planner. The released set of twisted speed and angular speed (or only angle) information will be acquired by the vehicle controller or wire control interface. Finally, the autonomous vehicle will be under controlled automatically.

2) Autoware low speed autonomous driving kit structure

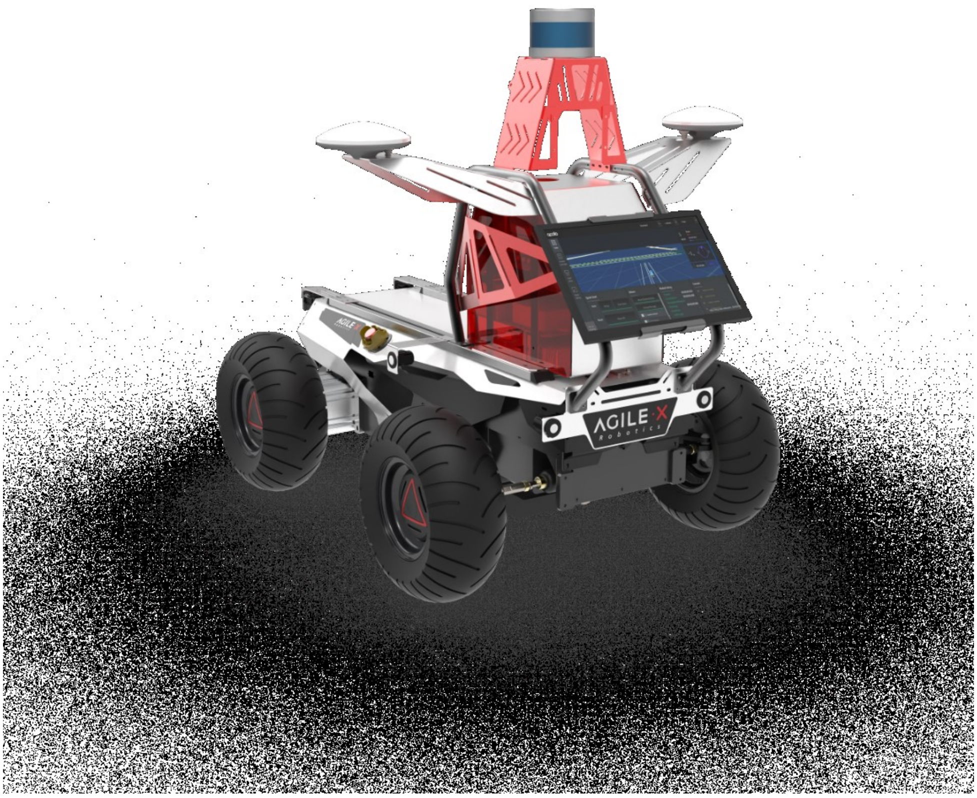
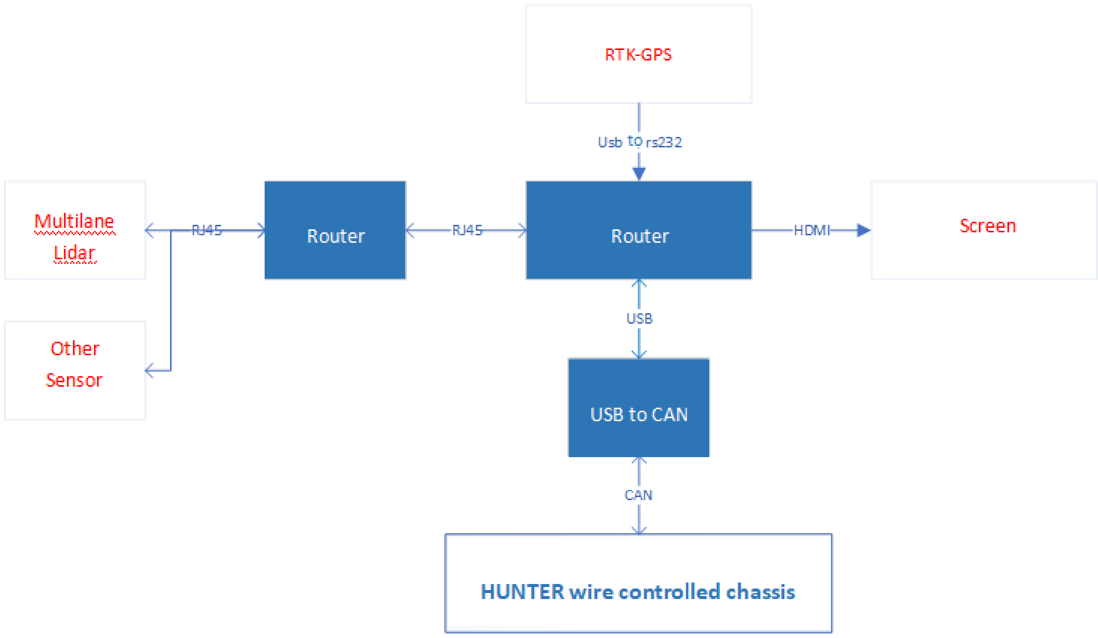


Figure 5 Vehicle platform chassis system data flow diagram



iii. Basic introduction of software

1) Basic introduction of ROS

ROS is Open Source robot operating system which based on Ubuntu system. It has highly flexible software architecture for robot software programming. This structure connects each node (independent program) in ROS, each node communicates based on TCP\IP, and the nodes are connected with each other through different themes. This structure includes a large number of tool software, code base and protocol which is aimed to simplify the difficulties and complexity of the process of creating complex and robust robot behaviors on the robot platform. It helps avoiding the re-creating wheels problem the which making the development easier and faster, skip the repetitive work.

ROS is an open source operating system for robot, it has all the functions that operating system do, including hardware abstraction, low-level device control, implementation of commonly used functionality,message-passing between processes, and package management. It also provides the services and library function for acquiring, editing and translating, complying and running code across computers. At the same time, ROS is also compatible with many third-party libraries including opencv (computer vision), PCL (point cloud library) and so on. The interface of ROS is also very diversity which is compatible with most sensors such as lidar, GPS, and ultrasonic on the market. Users can add various sensor devices to their robots based on their requirement. And ROS WIKI provides a large number of packages which developed by users , these open source packages can meet user’s professional project requirements. The main function of ROS is providing services designed for code reusing support for robot researching and development. ROS is a distributed process (node) frame, these process are encapsulated in the procedures packages and function that are easy to share and release. ROS also supports a joint system similar to code repository which can achieve project collaboration and released. This design can make the development and implementation of a project completely independent from the file system to the user interface (not restricted by ROS). At the same time, all projects can be integrated by ROS basic tools. In conclusion, ROS aims to make robot development become easier, faster and more interesting.

2) Basic introduction of Autoware

Autoware is the first integration open source software for autonomous driving vehicle in the world. Mostly, Autoware is suitable for cities, but highways, intersection areas and geo-fence are also applicable. All rights for Autoware's code base are reserved by is the Apache 2 license. For safety reasons, we provide a simulation environment based on ROSBAG for those who do not have self-driving technology.

(b) System software and hardware environment construction

1. Hardware installation

a) Accessories list

Accessories list	Table of Accessories list	Quantity of components	Remarks
Computing unit	Computing unit	1	
	Mouse and keyboard	1	
Multi line lidar	Multi line lidar sensor	1	
	Sensor controller	1	
Liquid crystal display module	Liquid crystal display screen	1	
	mini-hdmi to hdmi wire	1	
	usb to type-c wire	1	
usb-to-can	usb-to-can module	1	
Power module	24v to 12v	1	
	24v to 19v	1	
Chassis module	HUNTER mobile chassis	1	
	Aviation plug (with line)	1	
	Vehicle controller	1	

Note: Due to the different network standard in different countries and regions, unfortunately the router is not able to provide within the accessories list, users can purchase the router based on the requirement by themselves.

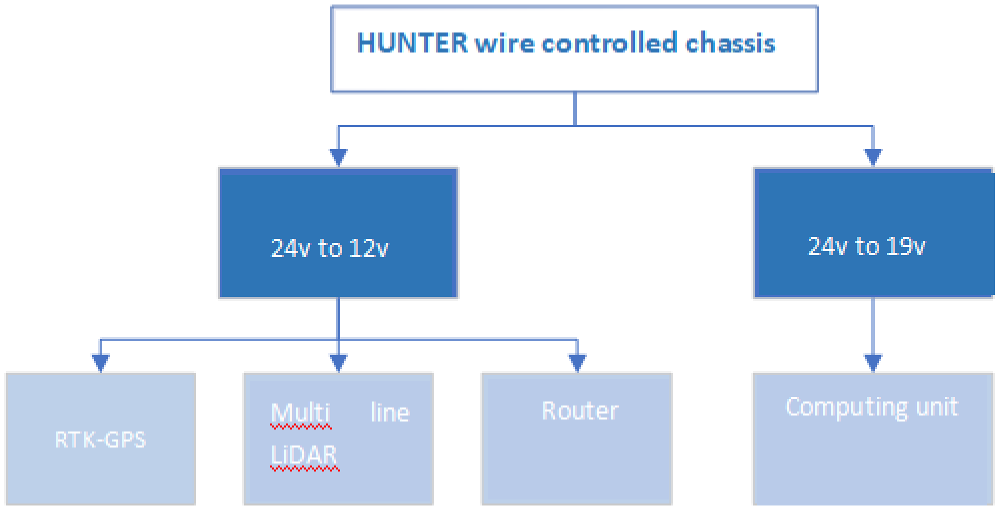
b) Accessories electrical description

Accessories name	Electrical characteristics of accessories	Remarks

Computing unit	DC 19v@6.5a	
Multi line lidar	DC 12v@0.8a (Typical values 9w)	
Liquid crystal display screen	DC 5v	
4G Router	DC 12v@0.8a	

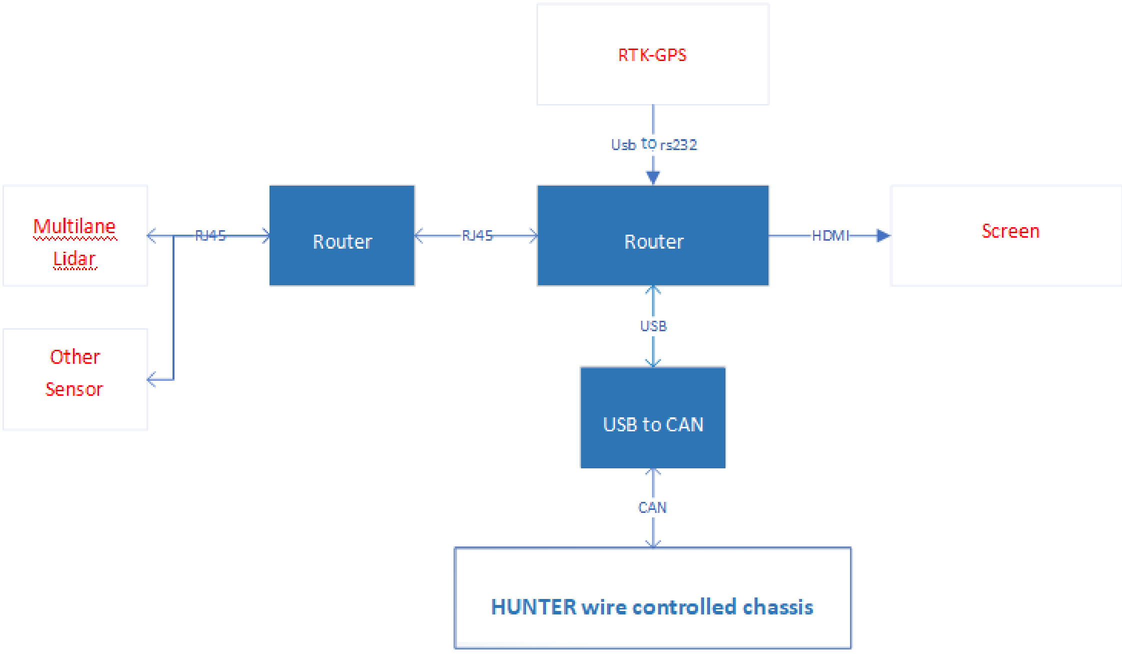
c) Power connection topographic diagram

The vehicle contains two voltage conversion modules, all of which are powered by the battery of the chassis, and both are from the aviation plug on the top of the hunter. The voltage is 21.5v~29.2v, and it will change with the external movement during use. However, the sensor module mentioned above mainly contains two electrical characteristics, one is 12v and the other is 19v, so two voltage stabilizing modules are used in the chassis.



d) Data flow diagram

The Autoware Autonomous Driving Education Development Kit contains the front-end perception, the intermediate data transmission and processing calculation, and the rear-end actuator. The front-end perception is consist of multi line lidar sensor and rtk-gps (adapting). In order to facilitate customers to develop new sensors, a 4G router is included in the accessories to facilitate customers to develop other sensor units. The data processing unit in the middle uses a intel i7 9700 processor, and it is equipped with a screen to debugging and use. The data flow diagram is shown in the following figure.



e) Installation

2. Software installation

ROS Installation, refer to: http://wiki.ros.org/kinetic/Installation/Ubuntu

```
$ sudo sh -c '. /etc/lsb-release && echo "deb
http://mirrors.ustc.edu.cn/ros/ubuntu/ `lsb_release -cs` main" >
/etc/apt/sources.list.d/ros-latest.list'


$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --
recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

$ sudo apt-get update

$ sudo apt-get install ros-kinetic-desktop-full

$ apt-cache search ros-kinetic

$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

$ source ~/.bashrc
$ sudo apt install python-rosdep python-rosinstall python-
rosinstall-generator python-wstool build-essential
$ sudo apt install python-rosdep
$ sudo rosdep init
```

Normally, there would be an error, then it is needed to modify the hosts file. Refer to: https://blog.csdn.net/u013468614/article/details/102917569

#Open hosts file

```
 sudo gedit /etc/hosts
```

#Add to the end of the file

151.101.84.133   raw.githubusercontent.com

#Exit after saving and then try again

```
$ sudo rosdep init
$ rosdep update
```

Create workspace:

```
$ mkdir -p ~/catkin_ws/src
$ cd catkin_ws/src/
$  catkin_init_workspace
$ cd ..
$ catkin_make
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Restart the computer.

Installation of Can analyzer rely:

Copy libcontrolcan.so to /usr/local/lib

```
$ sudo cp libcontrolcan.so /usr/local/lib
```

Can authorization Configuration ：

```
$ sudo gedit /etc/udev/rules.d/99-mysub.rules
```

Add contents:

##

ACTION=="add",SUBSYSTEMS==

"usb",ATTRS{idVendor}=="04d8",

ATTRS{idProduct}=="0053",

GROUP="users",MODE="0777"

When the configuration is completed

```
$ sudo ldconfig
```

Compile the code:

Copy hunter_robot in the src file folder to workspace

```
$ cd ~/catkin_ws/
$ catkin_make
```

Install qt: (Use 5.6.2 version here)

qt download page: https://www.qt.io/download

Go to the directory with the qt installation package:

```
$ sudo chmod +x qt-opensource-linux-x64-5.6.2.run
$ ./qt-opensource-linux-x64-5.6.2.run
```

Keep clicking Next until the installation is completed.

Install opencv:

Install 3.4.2 version here, interlink: https://opencv.org/releases/page/3/

Refer to: https://blog.csdn.net/u010632165/article/details/81387700

Unzip opencv, and then enter the opencv folder

```
$ sudo mkdir build
$ cd build
$ cmake ../
$ make -j8
$ sudo make install
```

Install Autoware, use 1.8.0 version here, source code:

https://gitlab.com/Autowarefoundation/Autoware.ai/Autoware/-/tree/1.8.0

If there is any problem with the installation process,. refer to:

https://blog.csdn.net/yourgreatfather/article/details/86504547

Unzip Autoware

```
$ cd Autoware-1.8.0/ros/
```

One-click installation of all relies:

```
$ rosdep install -y --from-paths src --ignore-src --rosdistro
$ROS_DISTRO
```

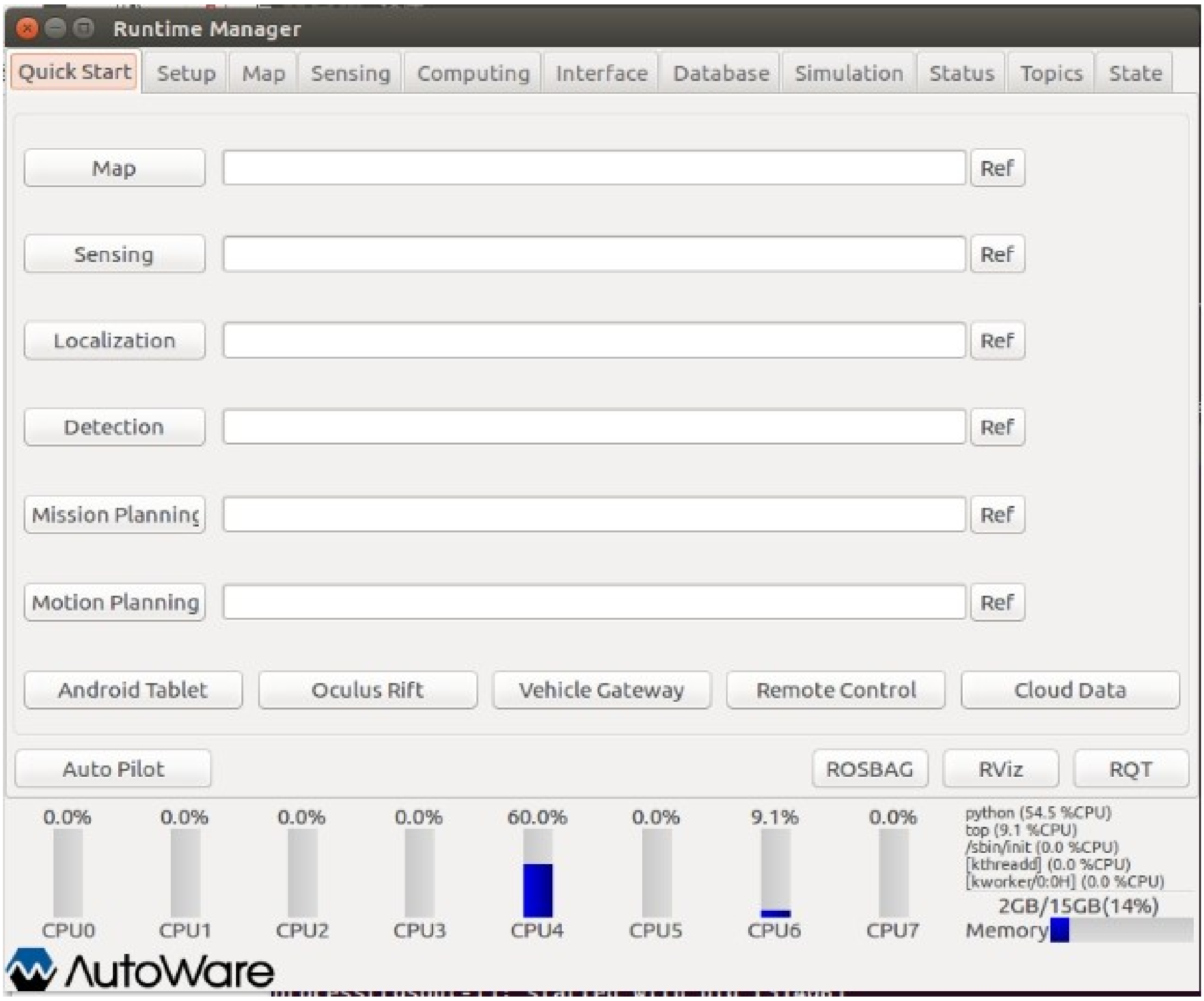
Compile

```
$ ./catkin_make_release
```

Start Autoware:

```
$ cd Autoware-1.8.0/ros/
$ ./run
```

After start up successfully, start up interface is shown as follow:



3. Vehicle wire control

With the technological development of automotive electronic and the integration of automotive systems, people can drive cars by means of electronic instead of traditional mechanical mechanisms to transmit control signals. This electronic technology is X-By-Wire. "By-Wire" can be called electronic wire control, and "X" represents various systems in the car. Such as steering-by-wire, Brake-By-Wire, etc.

Wire controlling is the basis of automatic driving. The basic requirement of wire controlling is changing all the control behaviors of the vehicle from mechanical to electronic control , from the original analog signal input to the digital signal input. Agilex Robotics chassis HUNTER provides services includes steer by wire, throttle by wire, and brake by wire. Besides the basic wire control function, our communication interface also provides some feedback information from chassis.

The CAN communication standard in HUNTER products uses the CAN2.0B standard, baud rate is 500K, message format is Motorola format. The linear velocity and the angular velocity of rotation of the chassis can be controlled through the external CAN bus interface. The information about current motion status and HUNTER chassis status would be given by HUNTER.

The protocol includes system status feedback frame, motion control feedback frame, and control frame. The content of the protocol as follows:

The system status feedback command includes the current car body status feedback, control mode status feedback, battery voltage feedback, and fault feedback. The protocol content is shown in Table 3.1.

Command Name	System Status Feedback Command			
Sending node	Receiving node	ID	Cycle(ms)	Receive-time out(ms)
Steer-by-wire chassis	Decision-making control unit	0×151	20ms	None
Data length	0×08			
Position	Function	Data type	Description	
byte[0]	Current status of vehicle body	unsigned int8	0×00 System in normal condition 0×01 Emergency stop mode(not enable) 0×01 System exception	
byte[1]	Mode control	unsigned int8	0×00 Remote control mode 0×01 Command control mode	
byte[2]	Battery voltage higher 8 bits	unsigned int16	Actual voltage X 10 (with an accuracy of 0.1V)	
byte[3]	Battery voltage lower 8 bits			
byte[4]	Failure information higher 8 bits	unsigned int16	See notes for details【**】	
byte[5]	Failure information lower 8 bits			
byte[6]	Count parity bit (count)	unsigned int8	0-255 counting loops,which will be added while single command has been sent	
byte[7]	Parity bit (checksum)	unsigned int8	Parity bit	

Description of Failure Information		
Byte	Bit	Meaning
byte [4]	bit [0]	Check error of CAN communication control command (0: No failure 1: Failure)
	bit [1]	Abnormal condition of front wheel steering encoder (0: No failure 1: Failure)
	bit [2]	RC transmitter disconnection protection (0: No failure 1: Failure)[1]
	bit [3]	Reserved, default 0
	bit [4]	Reserved, default 0
	bit [5]	Reserved, default 0
	bit [6]	Reserved, default 0
	bit [7]	Reserved, default 0
byte [5]	bit [0]	Battery under-voltage failure (0: No failure 1: Failure)
	bit [1]	Battery over-voltage failure (0: No failure 1: Failure)
	bit [2]	No.1 motor communication failure (0: No failure 1: Failure)
	bit [3]	No.2 motor communication failure (0: No failure 1: Failure)
	bit [4]	No.3 motor communication failure (0: No failure 1: Failure)
	bit [5]	No.4 motor communication failure (0: No failure 1: Failure)
	bit [6]	Motor drive over-temperature failure (0: No failure 1: Failure)
	bit [7]	Motor over-current failure (0: No failure 1: Failure)

The command of movement control feedback frame includes the feedback of current linear speed and angular speed of moving vehicle body. For the detailed content of protocol, please refer to Table 3.2.

Command Name	Movement control Feedback Command			
Sending node	Receiving node	ID	Cycle(ms)	Receive-time out(ms)
Steer-by-wire chassis	Decision-making control unit	0×131	20ms	None
Data length	0×08			
Position	Function	Data type	Description	
byte[0]	Moving speed higher 8 bits	signed int16	Actual speed X 100 (with an accuracy of 0.001rad)	
byte[1]	Moving speed lower 8 bits			
byte[2]	Internal steering angle higher 8 bits	signed int16	Actual speed X 100 (with an accuracy of 0.001rad)	
byte[3]	Internal steering angle lower 8 bits			

byte[4]	Reserved	-	0×00
byte[5]	Reserved	-	0×00
byte[6]	Count parity bit (count)	unsigned int8	0-255 counting loops,which will be added once every command sent
byte[7]	Parity bit (checksum)	unsigned int8	Parity bit

The control frame includes mode controlling, failure clearing command, control openness of linear speed, control openness of internal steering angle and sum check. For more protocol detail, please refer to Table 3.3.

Command Name	Control command			
Sending node	Receiving node	ID	Cycle(ms)	Receive-time out(ms)
Decision-making control unit	Chassis node	0×130	20ms	None
Data length	0×08			
Position	Function	Data type	Description	
byte[0]	Control mode	unsigned int8	0×00 Remote control mode 0×01 Command control mode[1]	
byte[1]	Failure clearing command	unsigned int8	See Note 2 for details*	
byte[2]	Linear speed percentage	signed int8	Maximum speed 1.50m/s, value range(-1,100)	
byte[3]	Internal steering angle percentage	signed int8	Maximum internal steering angle (-25°,25°), value range(-100,100)	
byte[4]	Reserved	-	0×00	
byte[5]	Reserved	-	0×00	
byte[6]	Count parity bit (count)	unsigned int8	0-255 counting loops,which will be added once every command sent	
byte[7]	Parity bit (checksum)	unsigned int8	Parity bit	

(c) Basic function demonstration and development tutorial

1) Early setting

lidar configuration, RS-LiDAR-16 as sample . Since Autoware is adapted to velodyne's lidar, it is necessary to modify the code of RS-LiDAR to have better adaption to Autoware.

Copy the ros_rslidar package to the workspace, refer to:
https://www.ncnynl.com/archives/201807/2552.html

There are two modifications. One is modifying the frame_id of the lidar and find the 27th line of code in ros_rslidar/rslidar_driver/src/rsdriver.cpp:
private_nh.param("frame_id", config_.frame_id, std::string("rslidar"));
change to:
private_nh.param("frame_id", config_.frame_id, std::string("velodyne"));

The other is modifying the output topic of lidar, find the 23rd line of code in ros_rslidar/rslidar_pointcloud/src/convert.cc:
private_nh.param("output_points_topic", output_points_topic, std::string("rslidar_points"));
change into:
private_nh.param("output_points_topic", output_points_topic, std::string("points_raw"));

After saving, enter the workspace and compile:

```
$ cd ~/catkin_ws
$ catkin_make
```

Change lidar IP, click system setting→Network→Cable→Option→IPv4 setting

Set the IP to 192.168.1.102, restart computer after the setting is completed.

2) Vehicle status feedback, control the vehicle through the keyboard

Course 1: Start the chassis and control

```
$ roscore
$ rosrun hunter_robot hunter_robot
```

Control by the keyboard:
Install the package teleop_twist_keyboard, refer to:
https://blog.csdn.net/allians/article/details/80583652

Download

```
$ sudo apt-get install ros-kinetic-teleop-twist-keyboard
```

Start

```
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

The speed should not be too fast, reduce the speed to around 0.2m/s by pressing the Z key on the keyboard. And then the buttons u l o j k l m are for controlling the chassis.

Controlling by the handle:
Install the package joy_node, refer to:
https://blog.csdn.net/han_l/article/details/77885238

Download

```
$ sudo apt-get install ros-kinetic-joy
```

Start

```
$ roslaunch hunter_robot joy.launch
```

Now you can control the chassis by Bluetooth handle.

The Bluetooth handle module is BETOP (Taobao 200 yuan), it is better to put a picture of the handle.
2), 3D laser point cloud data acquirement
There are two ways to create maps which are online map creation and offline map creation. We use offline map creation mostly , because the result of online map creation is a bit disappointed.

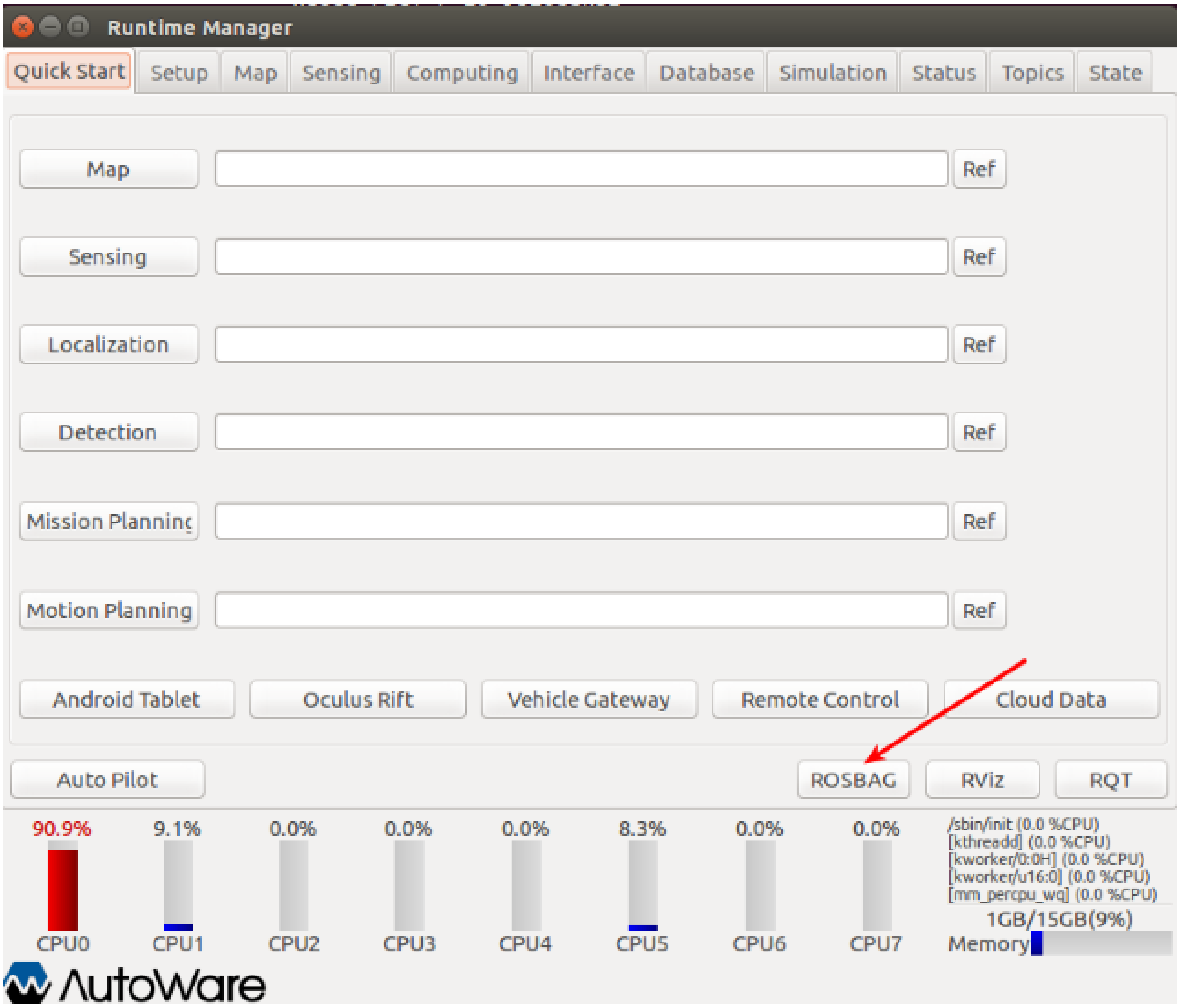
Start Autoware:

```
$ cd Autoware-1.8.0/ros/
$ ./run
```

Start lidar:

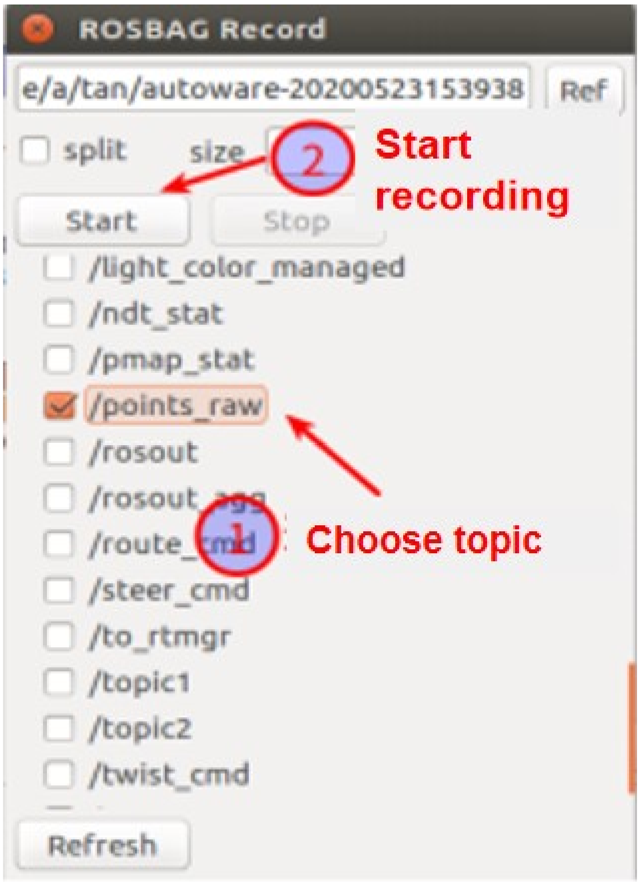
```
$ roslaunch rslidar_pointcloud rs_lidar_16.launch
```

Click the ROSBAGd on Autoware interface

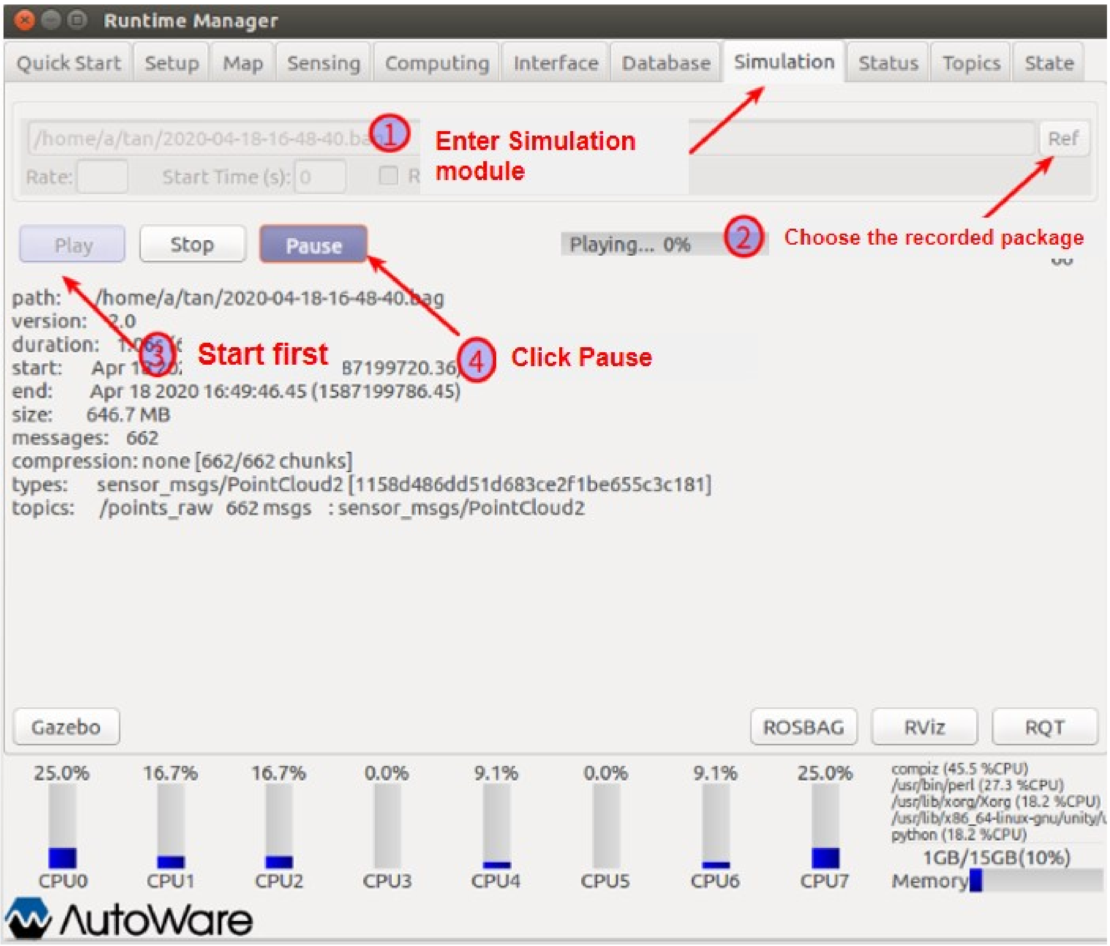


Select the lidar data/points_raw which need to be recorded, click Start to start recording

Controlling the vehicle to run a round in an unknown environment, try to move as slow as possible, remember the location of the starting point which is needed use for tracking along the trail. After recording is completed, click Stop to stop recording, and put the recorded package in a folder.

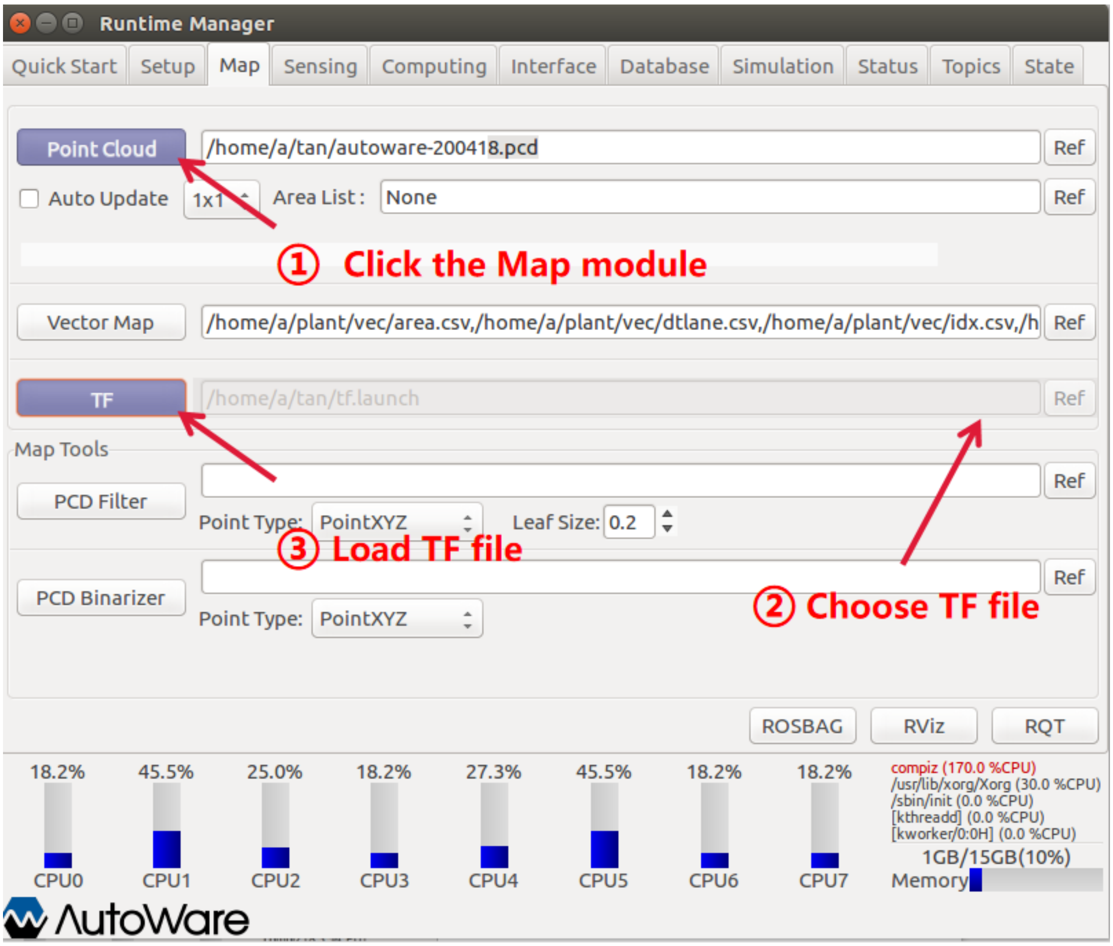


Enter the Simulation module of Autoware, select the package that just recorded, click Play and click Pause to pause immediately.

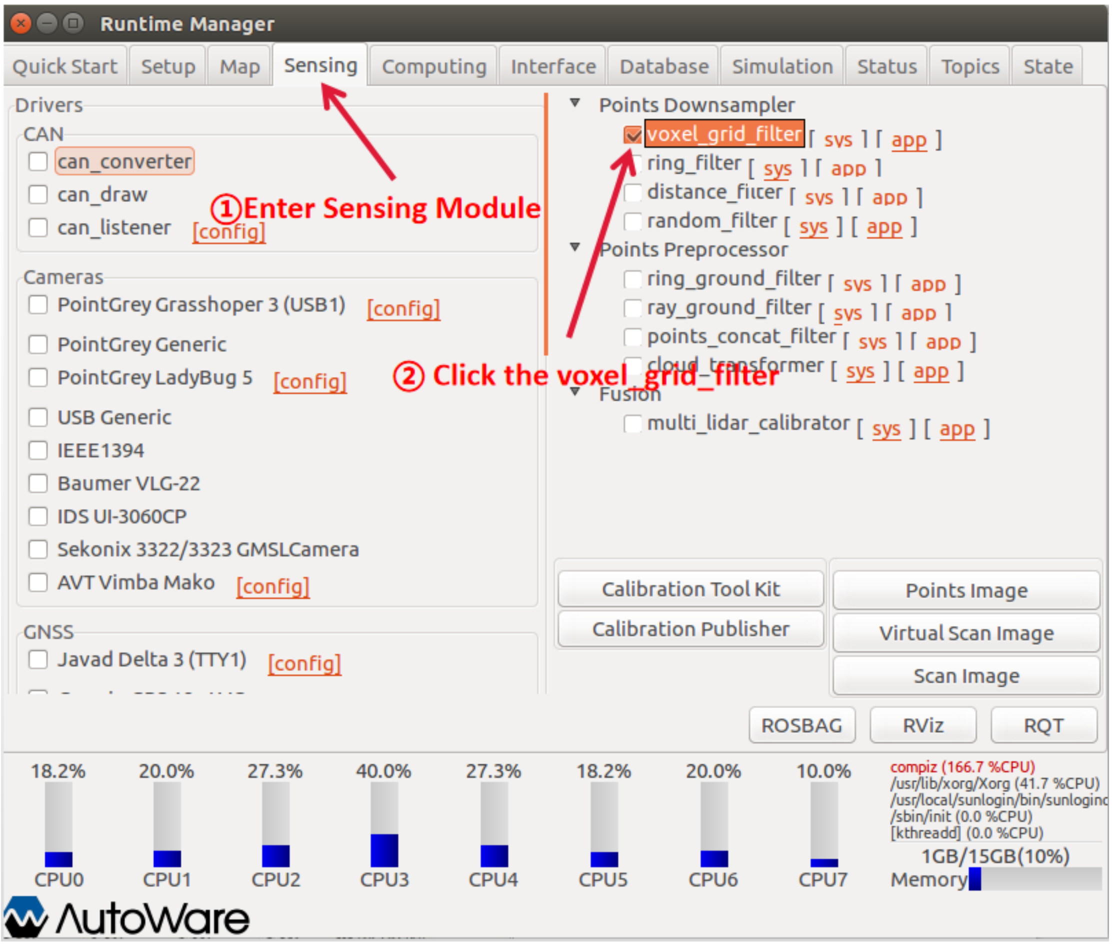


Enter the map module of Autoware, select TF, here is a reference tf.launch which aim to connect the world coordinate system with the map coordinate system, and the base_link coordinate system with the velodyne coordinate system.

```
<!---->
<launch>
<node pkg="tf"   type="static_transform_publisher" name="world_to_map" args="0 0 0 0 0 0 /world /map 10" />
<node pkg="tf"   type="static_transform_publisher" name="base_link_to_velodyne" args="0 0 0 0 0 0 /base_link /velodyne   10" />
</launch>
```

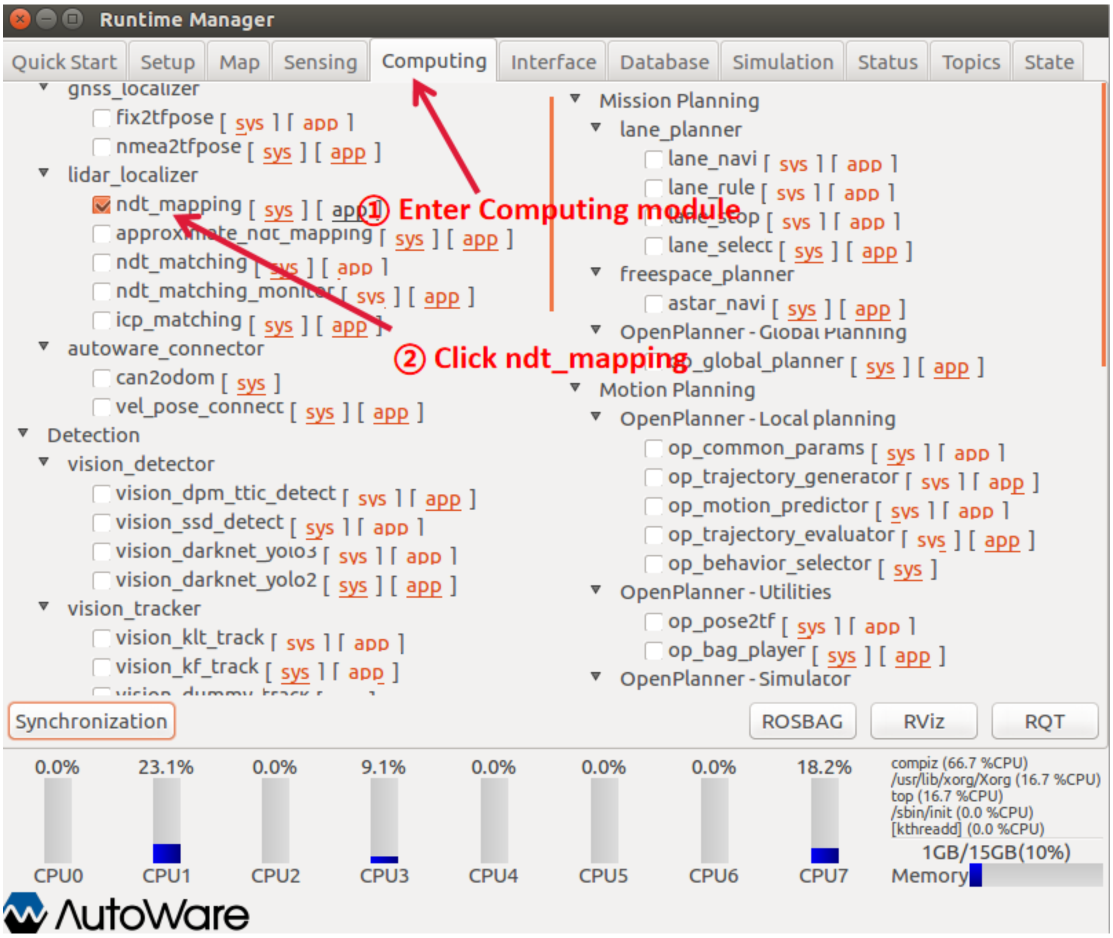


Enter Sensing module, click voxel_grid_filter function of Points Downsampler, this function is the filtration of lidar data.

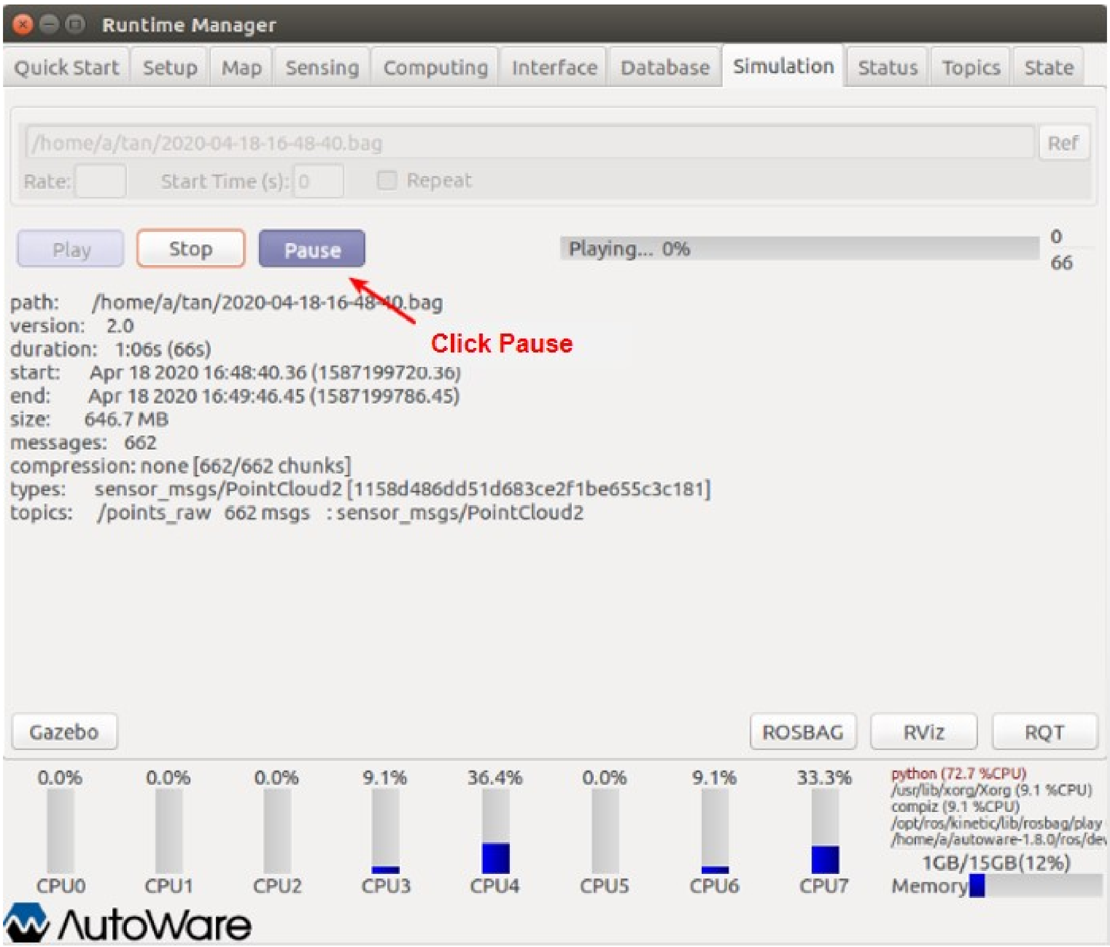


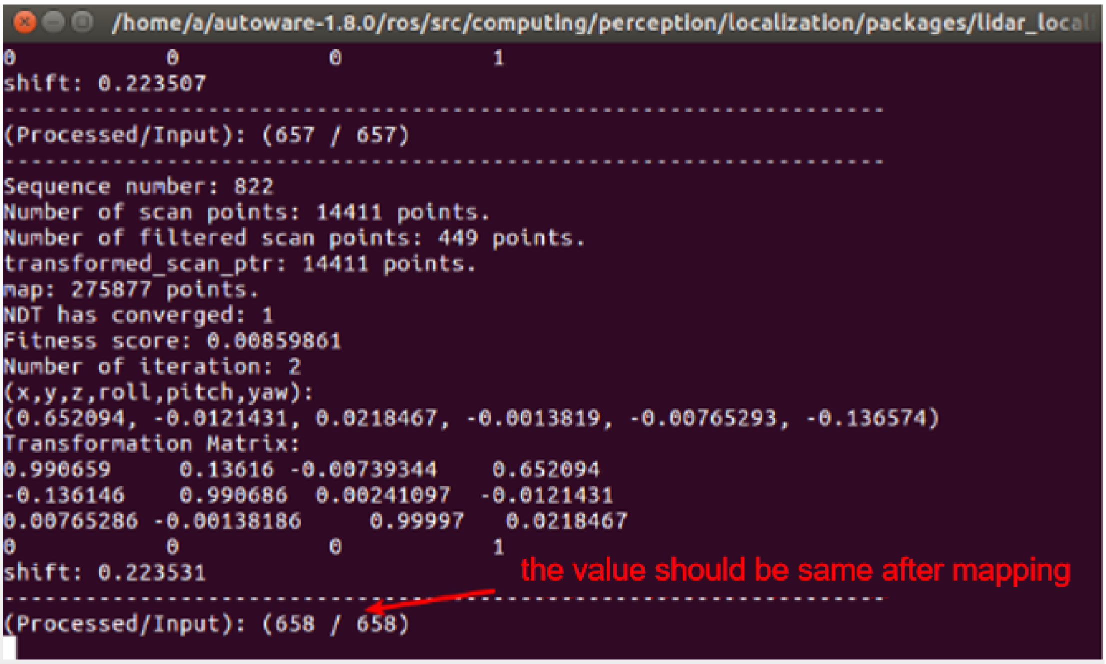
Enter Computing module

Click localization->lidar_localizer->ndt_mapping

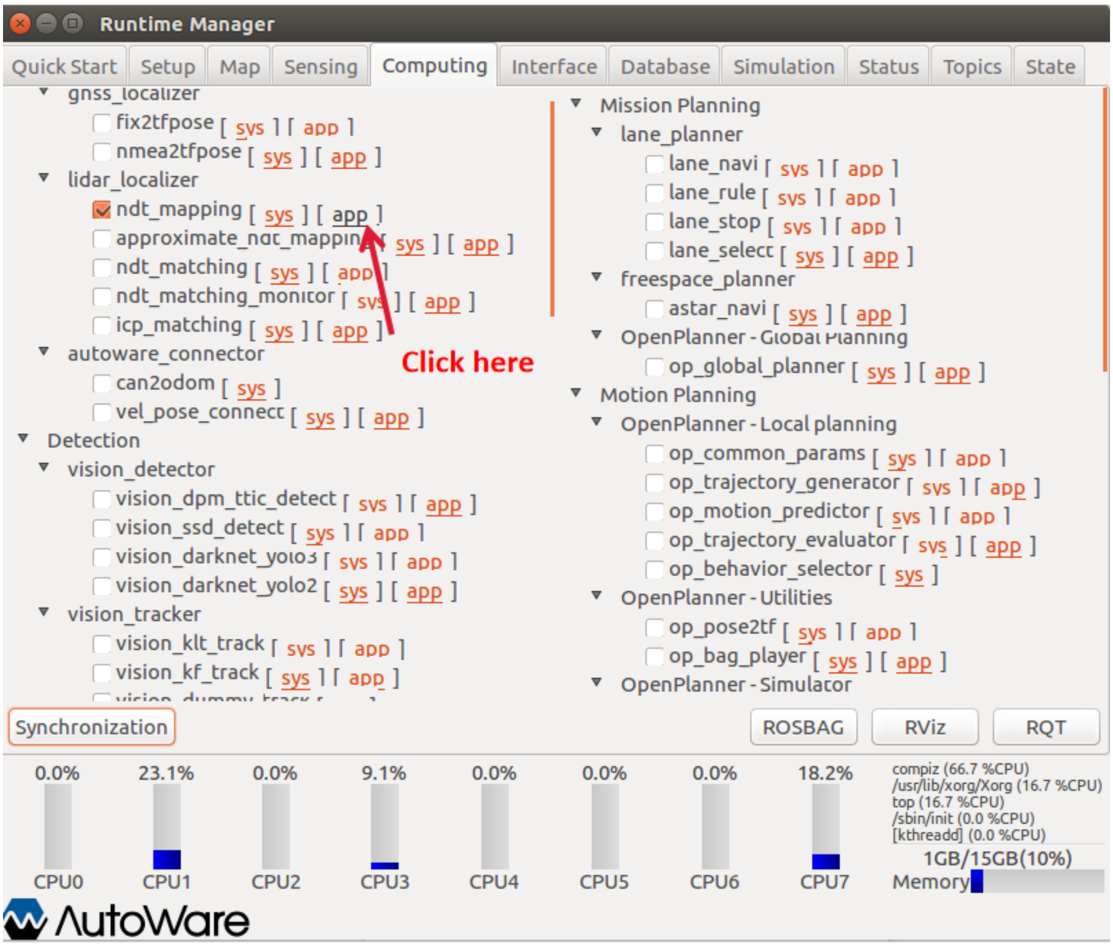


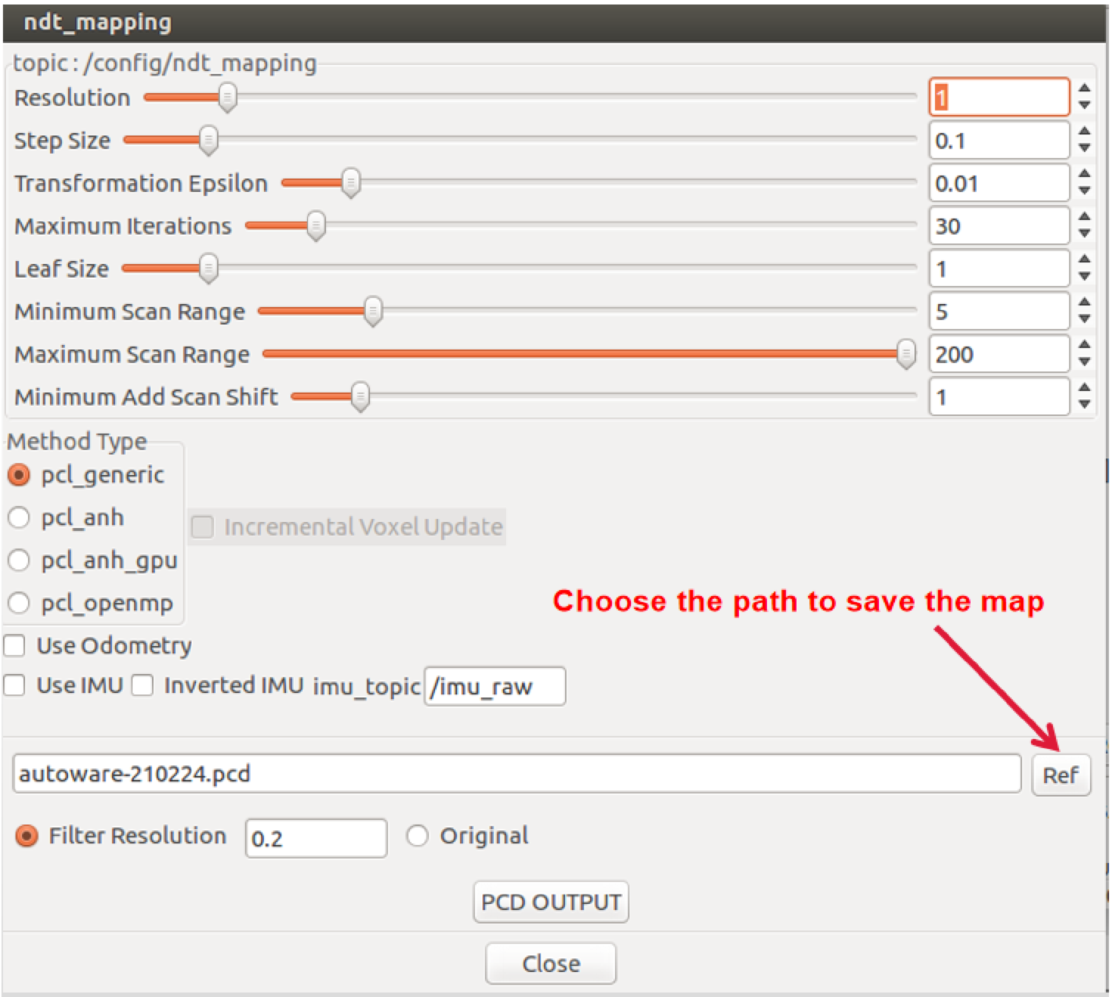
You can see the progress of the map creation in the terminal:

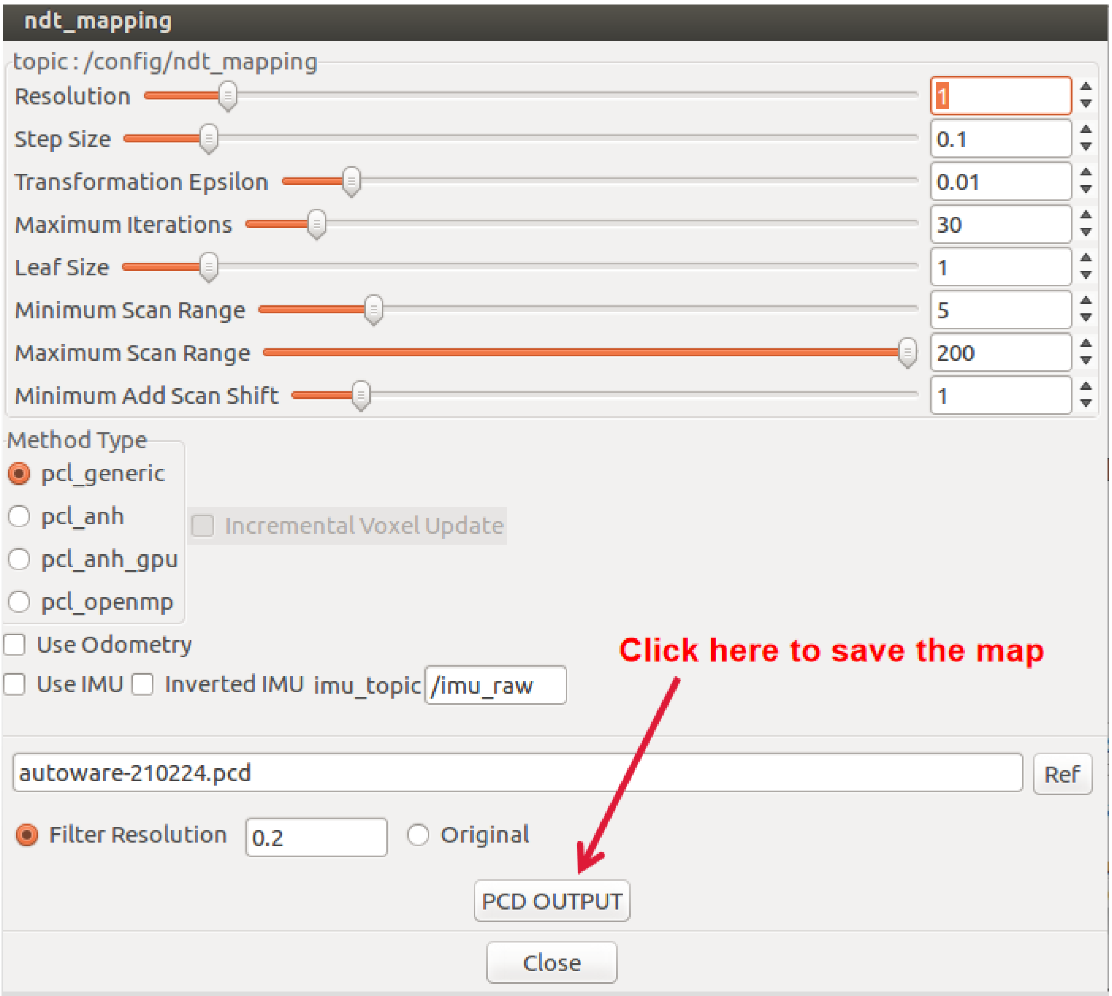




Back to Computing module


Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Topics
State
gnss_localizer
fix2tfpose [sys] [app]
nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys] [app]
ndt_matching [sys] [app]
ndt_matching_monitor [sys] [app]
icp_matching [sys] [app]
autoware_connector
can2odom [sys]
vel_pose_connect [sys] [app]
Detection
vision_detector
vision_dpm_ttic_detect [sys] [app]
vision_ssd_detect [sys] [app]
vision_darknet_yolo3 [sys] [app]
vision_darknet_yolo2 [sys] [app]
vision_tracker
vision_klt_track [sys] [app]
vision_kf_track [sys] [app]
Click here
Mission Planning
lane_planner
lane_navi [sys] [app]
lane_rule [sys] [app]
lane_stop [sys] [app]
lane_select [sys] [app]
freespace_planner
astar_navi [sys] [app]
OpenPlanner - Global Planning
op_global_planner [sys] [app]
Motion Planning
OpenPlanner - Local planning
op_common_params [sys] [app]
op_trajectory_generator [sys] [app]
op_motion_predictor [sys] [app]
op_trajectory_evaluator [sys] [app]
op_behavior_selector [sys]
OpenPlanner - Utilities
op_pose2tf [sys] [app]
op_bag_player [sys] [app]
OpenPlanner - Simulator
Synchronization
ROSBAG
RViz
RQT
0.0%
23.1%
0.0%
9.1%
0.0%
0.0%
0.0%
18.2%
CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
compiz (66.7 %CPU)
/usr/lib/xorg/Xorg (16.7 %CPU)
top (16.7 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
1GB/15GB(10%)
Memory
AutoWare


ndt_mapping
topic : /config/ndt_mapping
Resolution
1
Step Size
0.1
Transformation Epsilon
0.01
Maximum Iterations
30
Leaf Size
1
Minimum Scan Range
5
Maximum Scan Range
200
Minimum Add Scan Shift
1
Method Type
pcl_generic
pcl_anh
pcl_anh_gpu
pcl_openmp
Incremental Voxel Update
Choose the path to save the map
Use Odometry
Use IMU
Inverted IMU
imu_topic
/imu_raw
autoware-210224.pcd
Ref
Filter Resolution
0.2
Original
PCD OUTPUT
Close

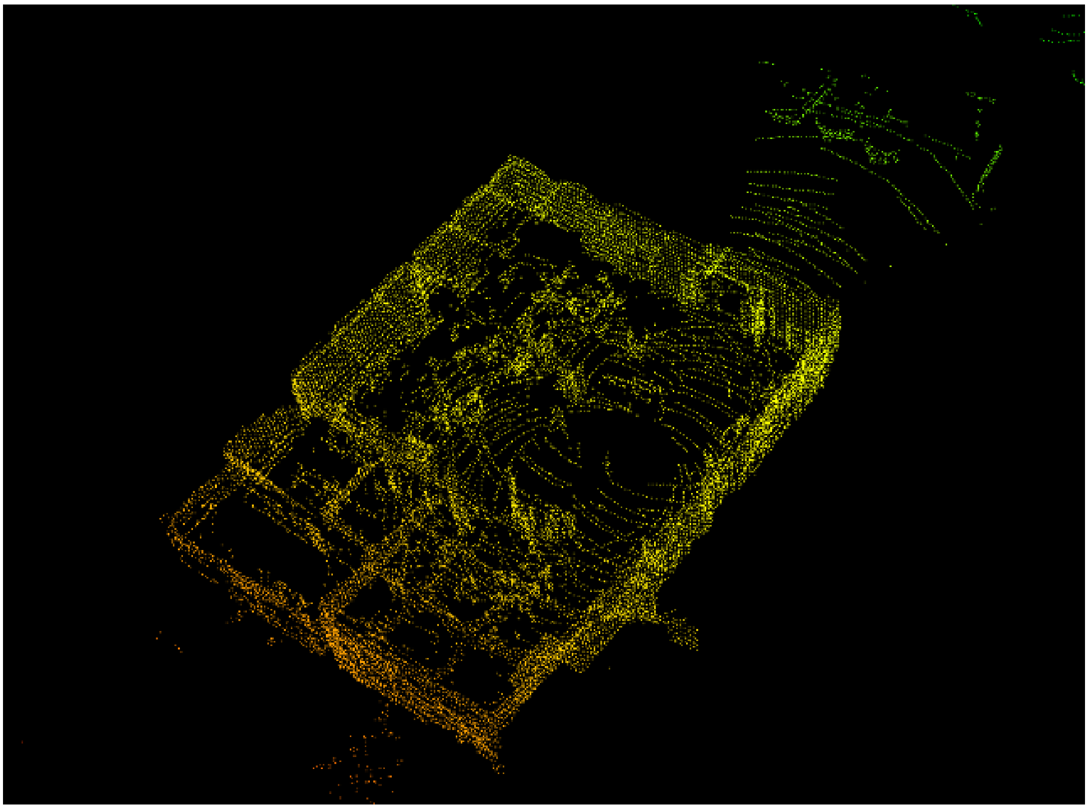


If you want to view the map, install

```
$ sudo apt-get install pcl-tools
```

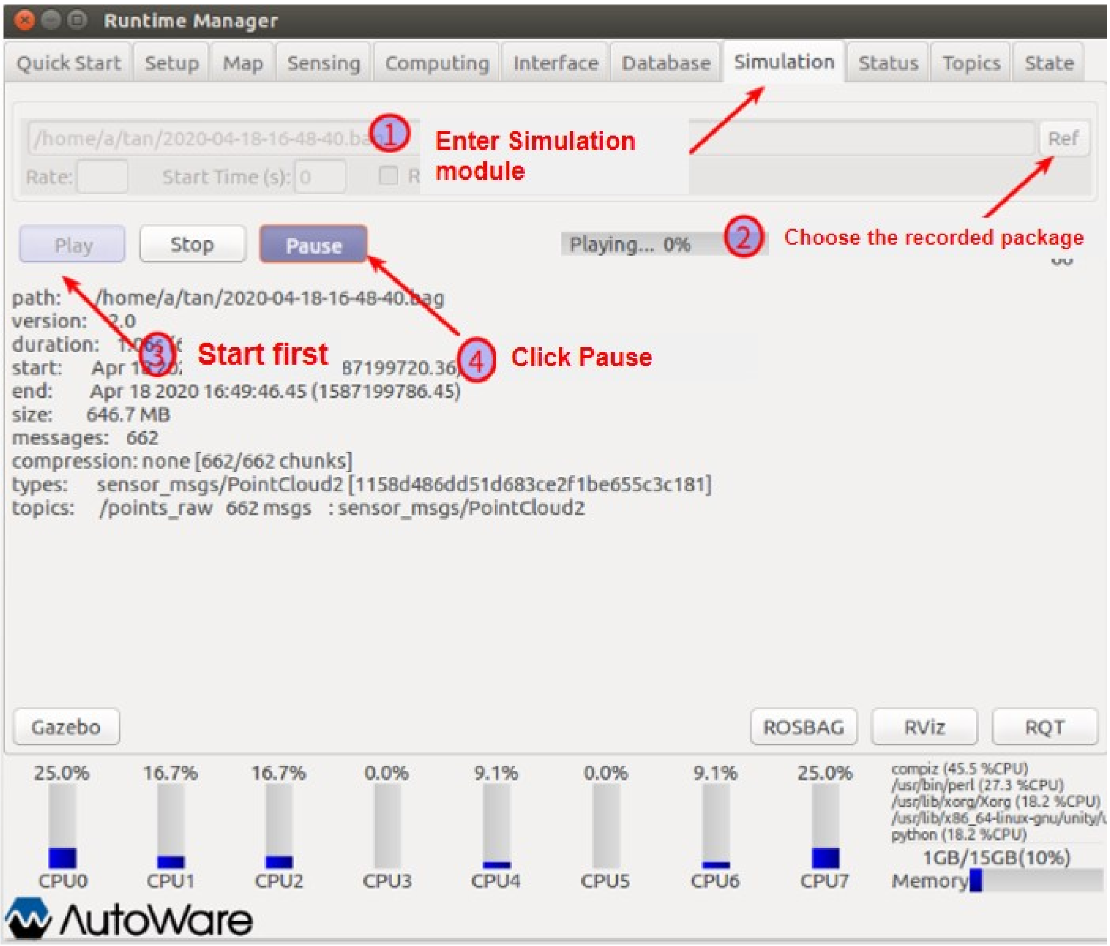
Then you can view the map, Press the numbers 1 2 3 4 on the keyboard to change the map color

```
$ pcl_viewer Autoware-200418.pcd
```

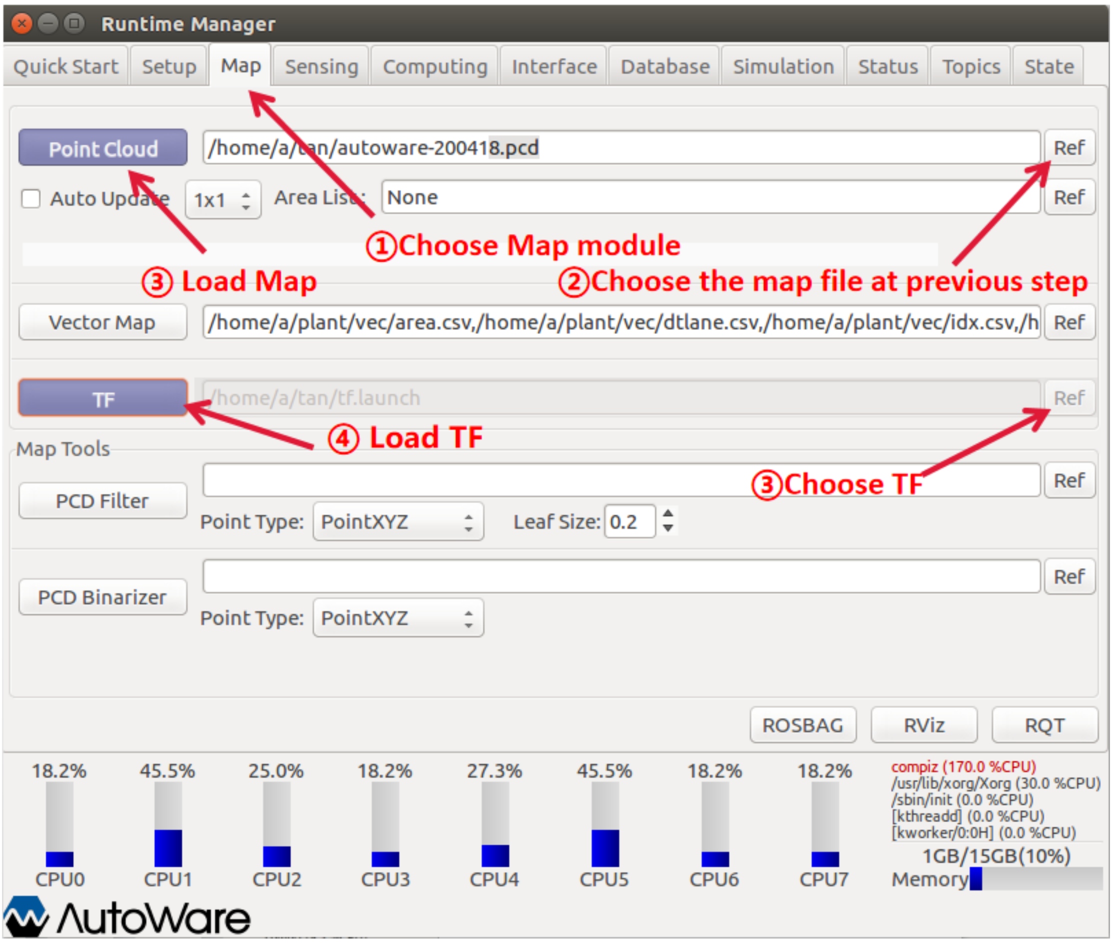


3) Demonstration of mapping, waypoint recording, waypoint following function

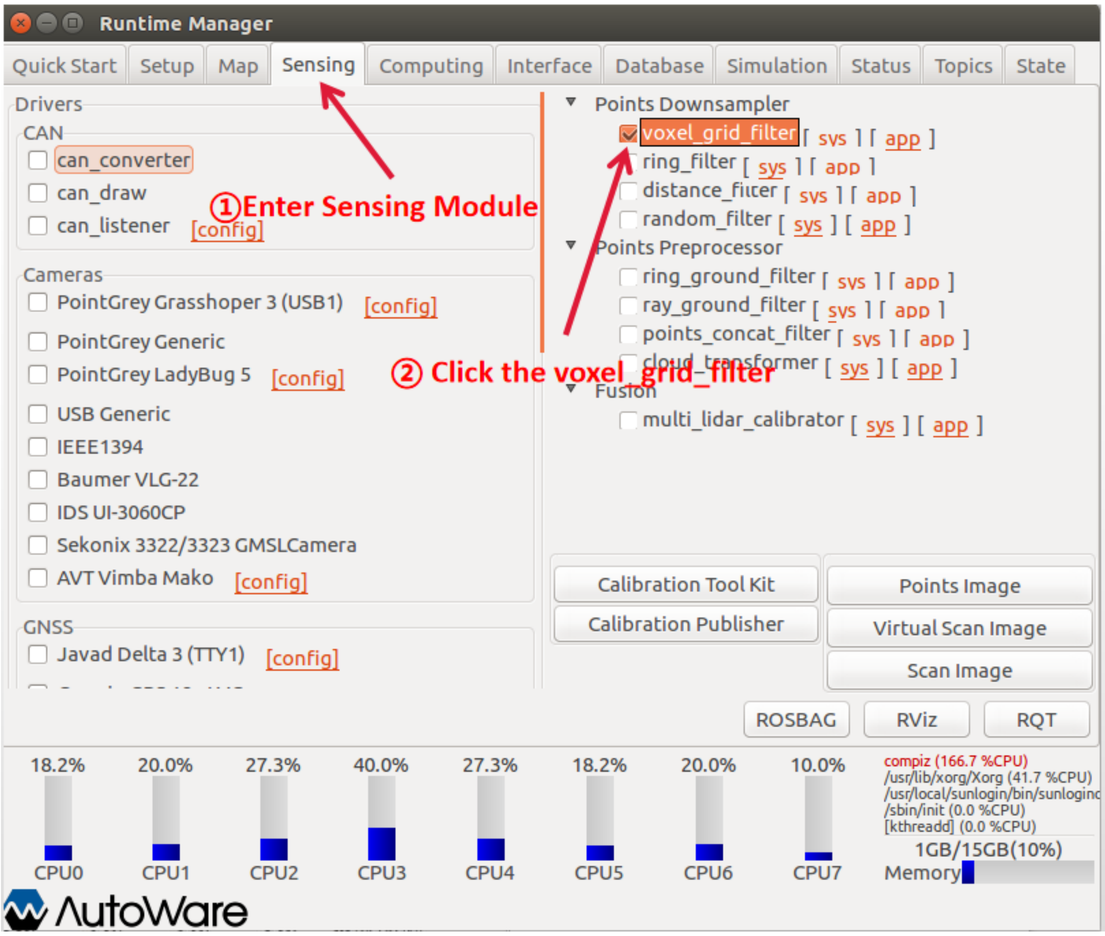
The waypoint recording is also offline, first enter the Simulation module



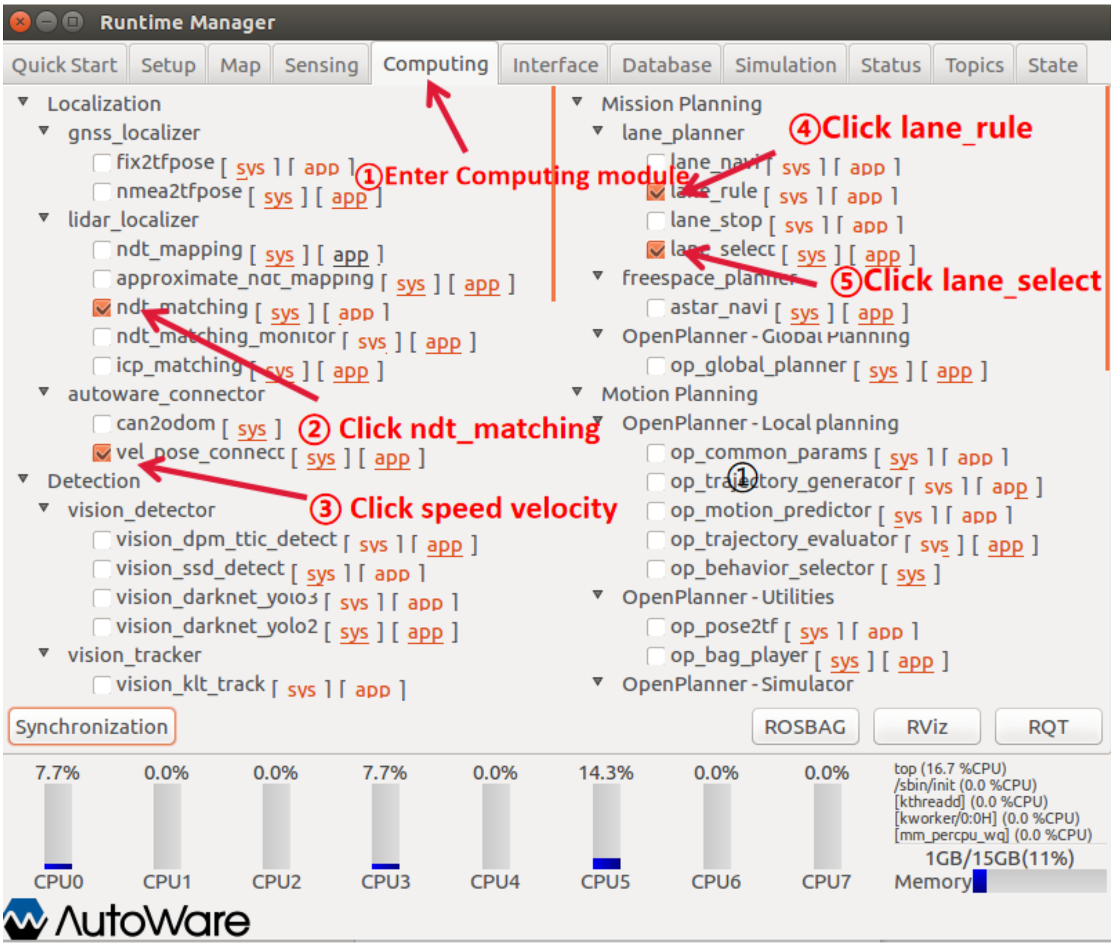
Enter Map module

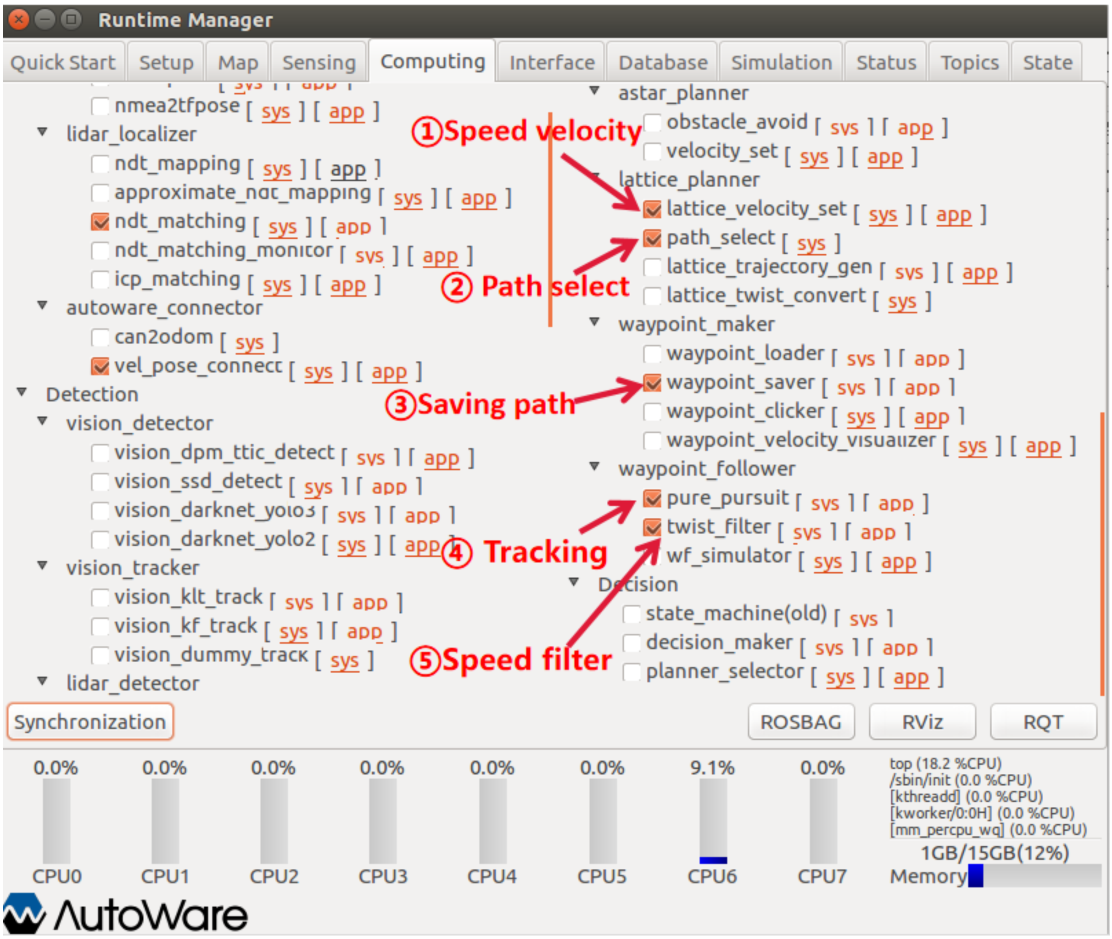


Enter Sensing module

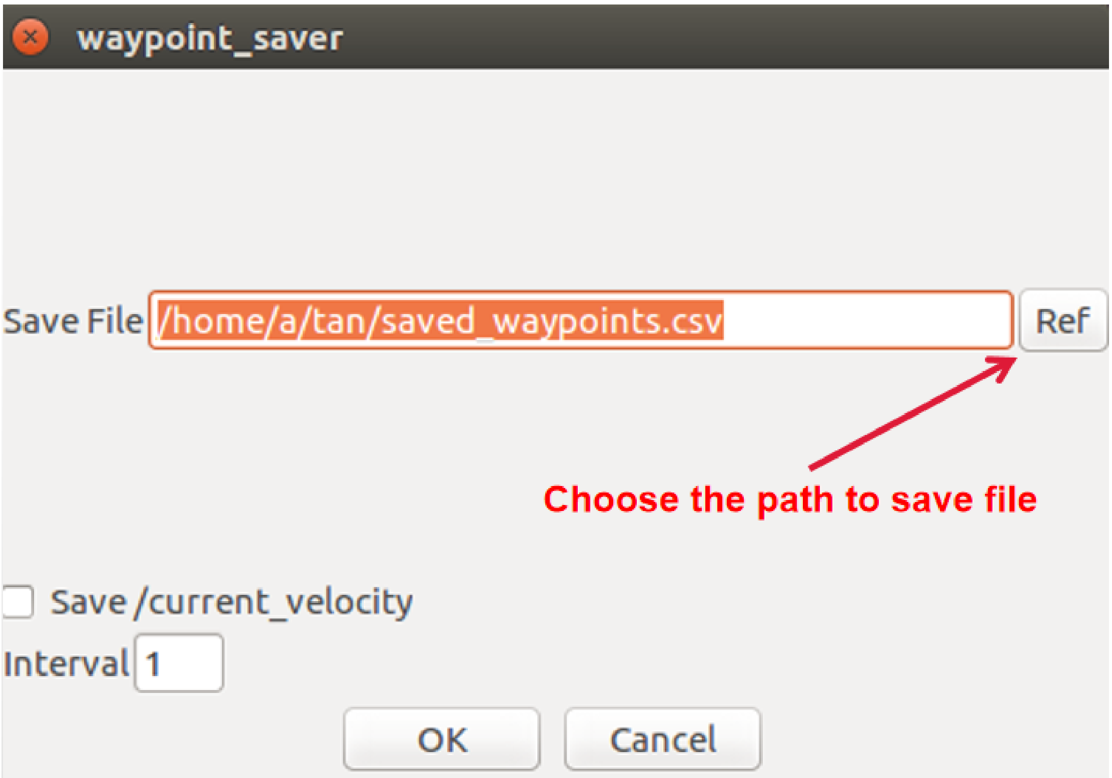


Enter Computing module

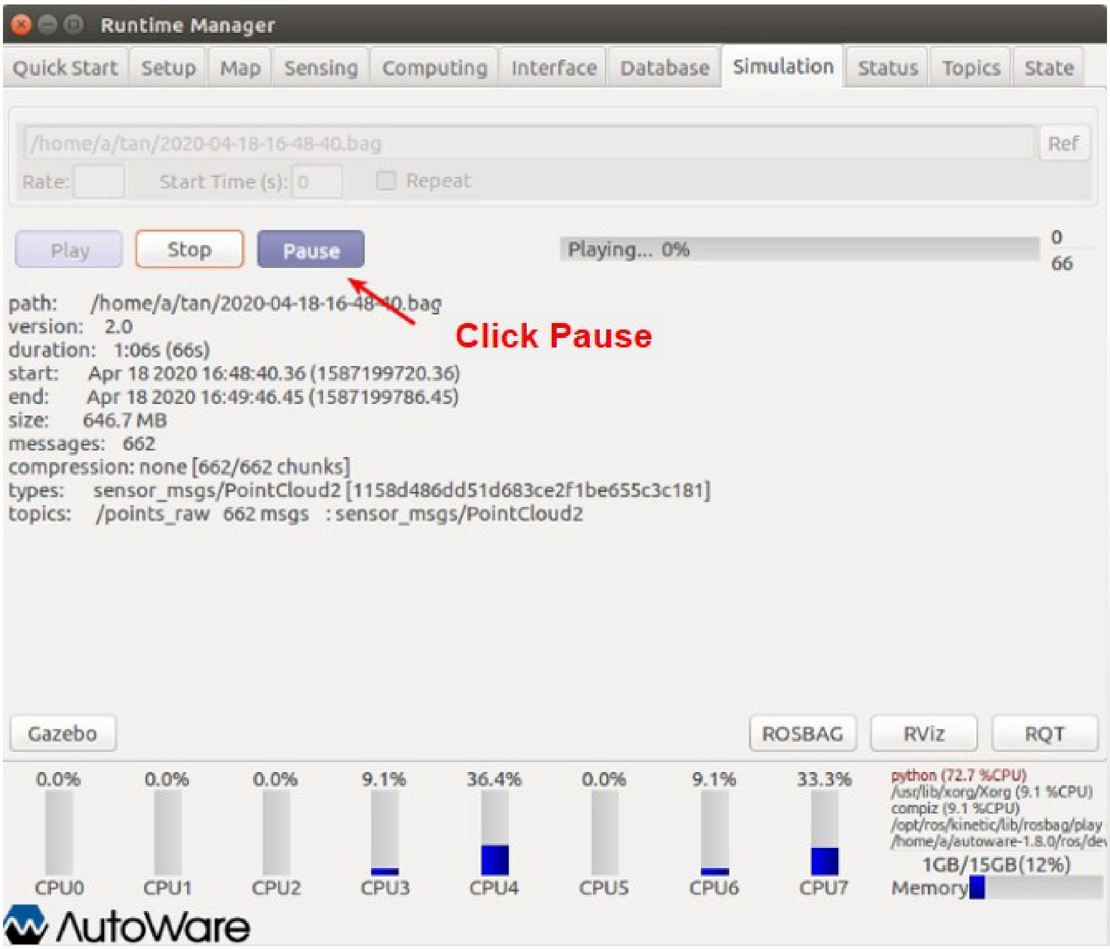

Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Topics
State
①Enter Computing module
② Click ndt_matching
③ Click speed velocity
④Click lane_rule
⑤Click lane_select
Localization
gnss_localizer
fix2tfpose [sys] [app]
nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys] [app]
ndt_matching [sys] [app]
ndt_matching_monitor [sys] [app]
icp_matching [sys] [app]
autoware_connector
can2odom [sys]
vel_pose_connect [sys] [app]
Detection
vision_detector
vision_dpm_ttic_detect [sys] [app]
vision_ssd_detect [sys] [app]
vision_darknet_yolo3 [sys] [app]
vision_darknet_yolo2 [sys] [app]
vision_tracker
vision_klt_track [sys] [app]
Mission Planning
lane_planner
lane_navi [sys] [app]
lane_rule [sys] [app]
lane_stop [sys] [app]
lane_select [sys] [app]
freespace_planner
astar_navi [sys] [app]
OpenPlanner - Global Planning
op_global_planner [sys] [app]
Motion Planning
OpenPlanner - Local planning
op_common_params [sys] [app]
op_trajectory_generator [sys] [app]
op_motion_predictor [sys] [app]
op_trajectory_evaluator [sys] [app]
op_behavior_selector [sys]
OpenPlanner - Utilities
op_pose2tf [sys] [app]
op_bag_player [sys] [app]
OpenPlanner - Simulator
Synchronization
ROSBAG
RViz
RQT
7.7% 0.0% 0.0% 7.7% 0.0% 14.3% 0.0% 0.0%
CPU0 CPU1 CPU2 CPU3 CPU4 CPU5 CPU6 CPU7
top (16.7 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0H] (0.0 %CPU)
[mm_percpu_wq] (0.0 %CPU)
1GB/15GB(11%)
Memory
AutoWare



In the third step, then select a folder to save the path file path waypoint_saver.



Back to the Simulation module, click Pause to start recording the path file, and wait for the bag finished. Then you can generate a saved_waypoints.csv file.



4) Constructing 3D point cloud map with Autoware and view 3D point cloud data

Firstly, it is needed to record the point cloud data package:

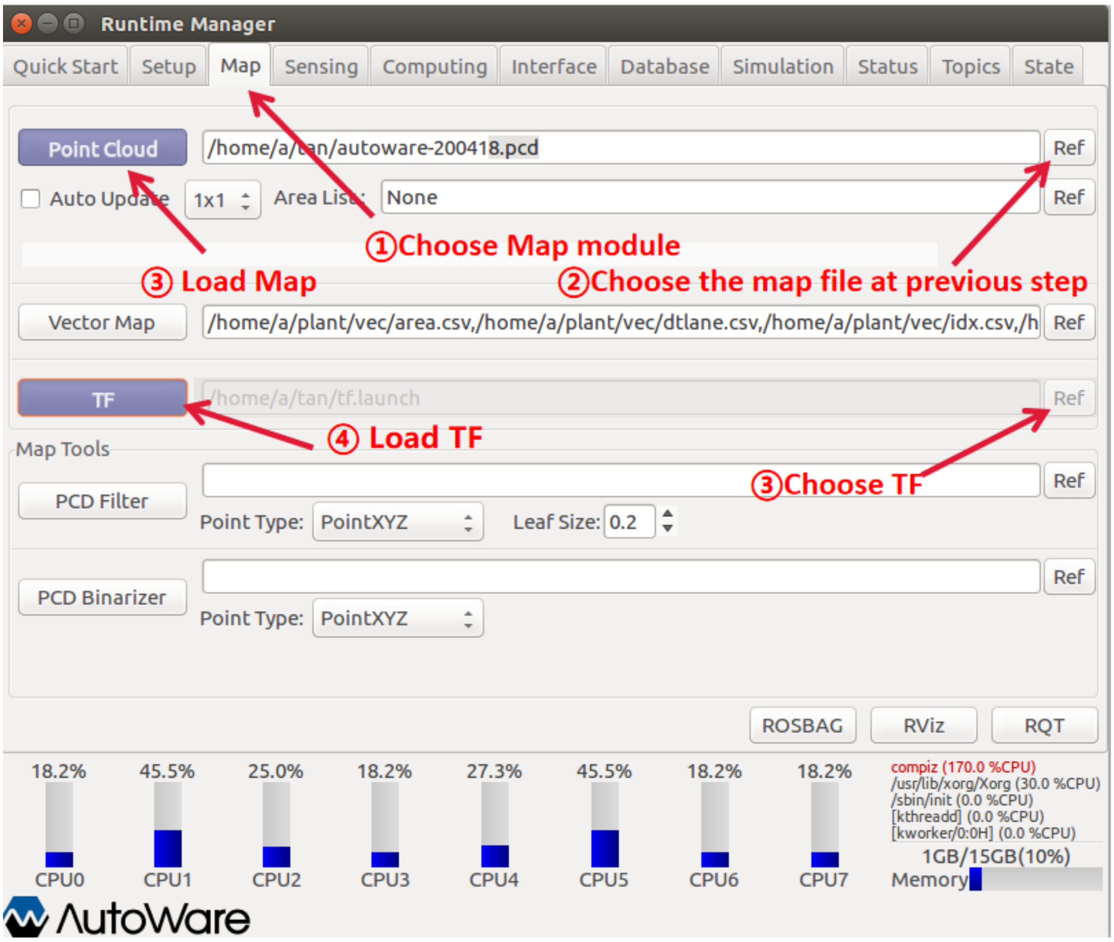
Start lidar:

```
$ roslaunch rslidar_pointcloud rs_lidar_16.launch
```

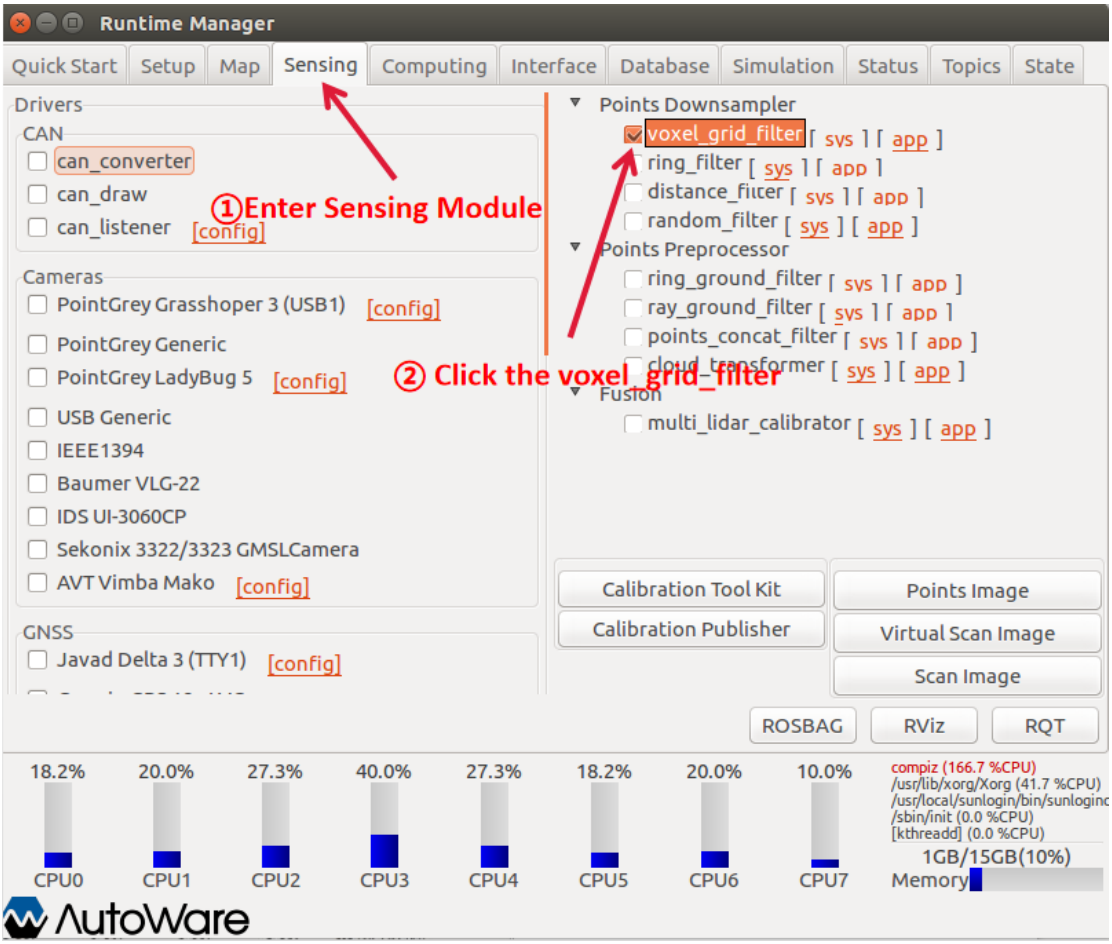
Start Autoware:

```
$ cd Autoware-1.8.0/ros/
$ ./run
```

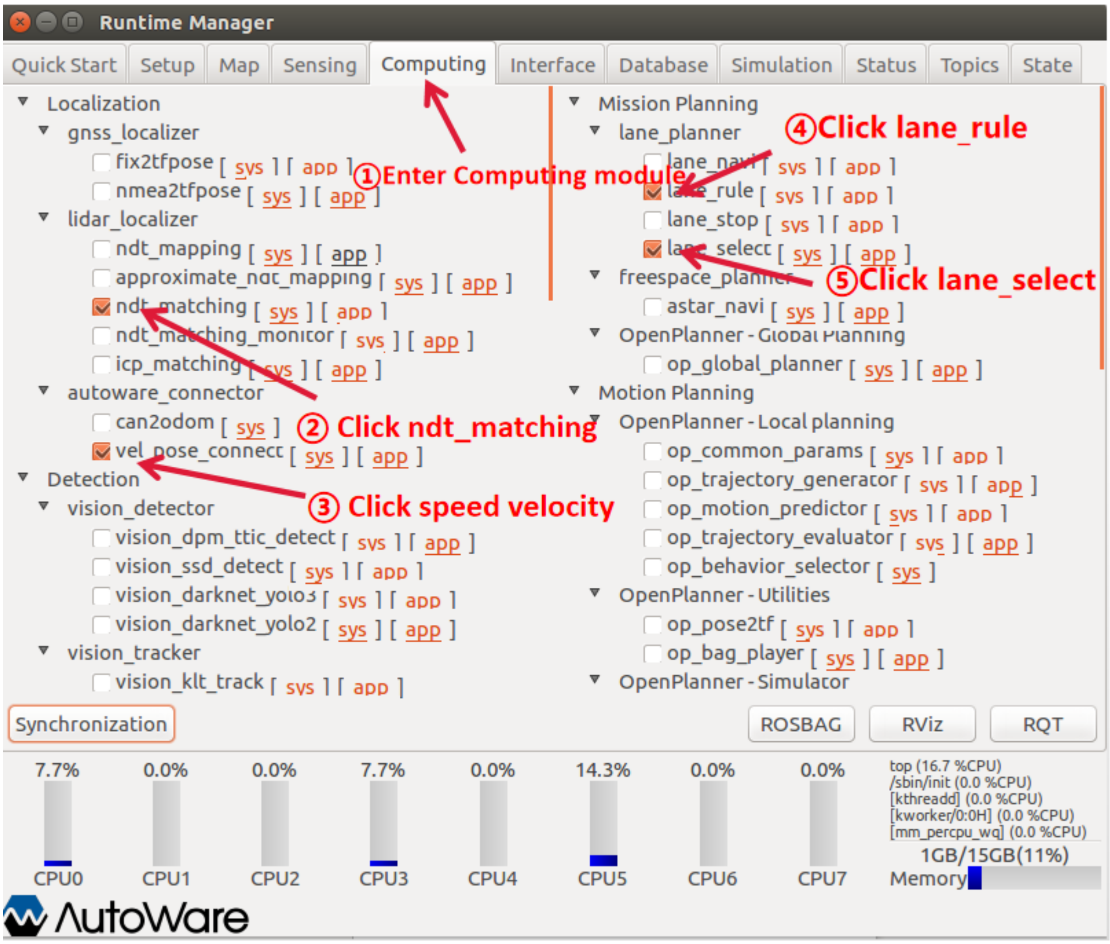
Enter Map module, loading map and TF



Enter Sensing module, loading point cloud filtering



Enter Computing module



Starting the chassis:

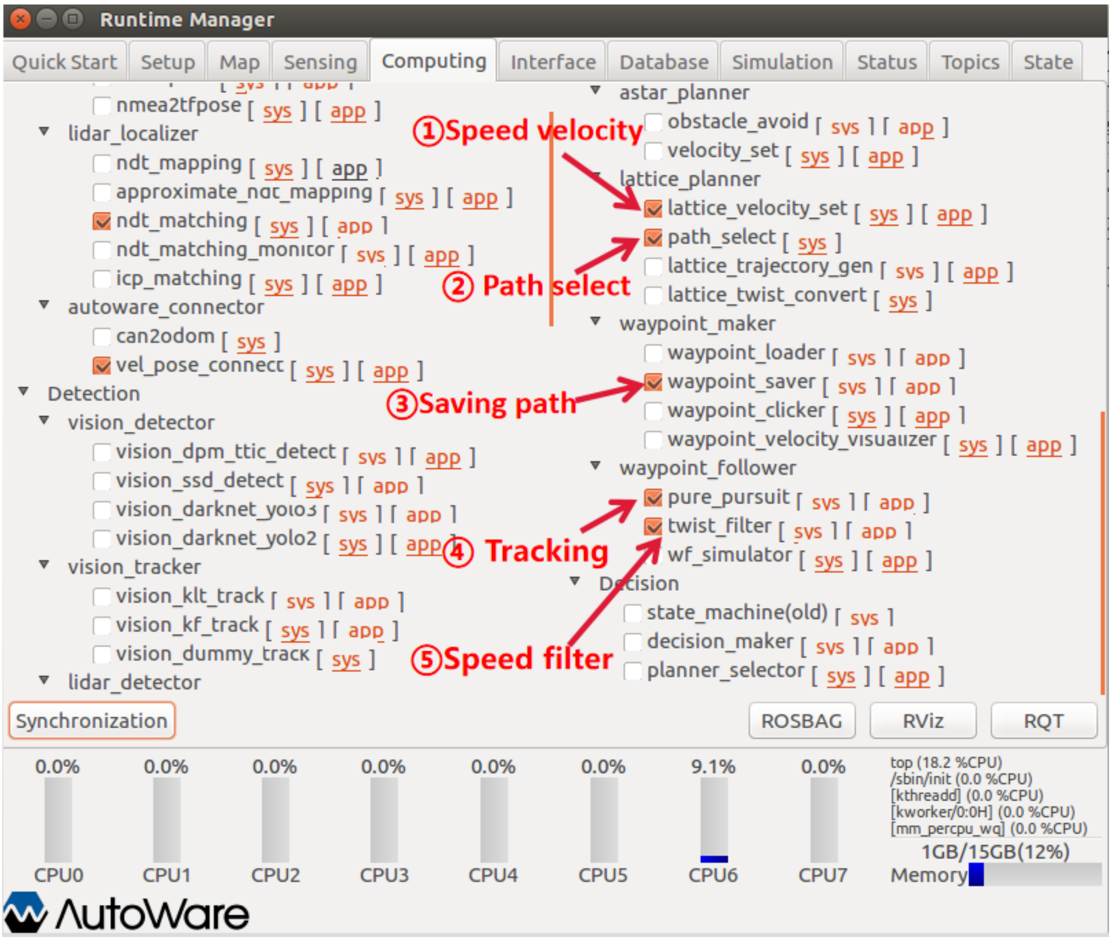
$ rosrun hunter_robot hunter_robot

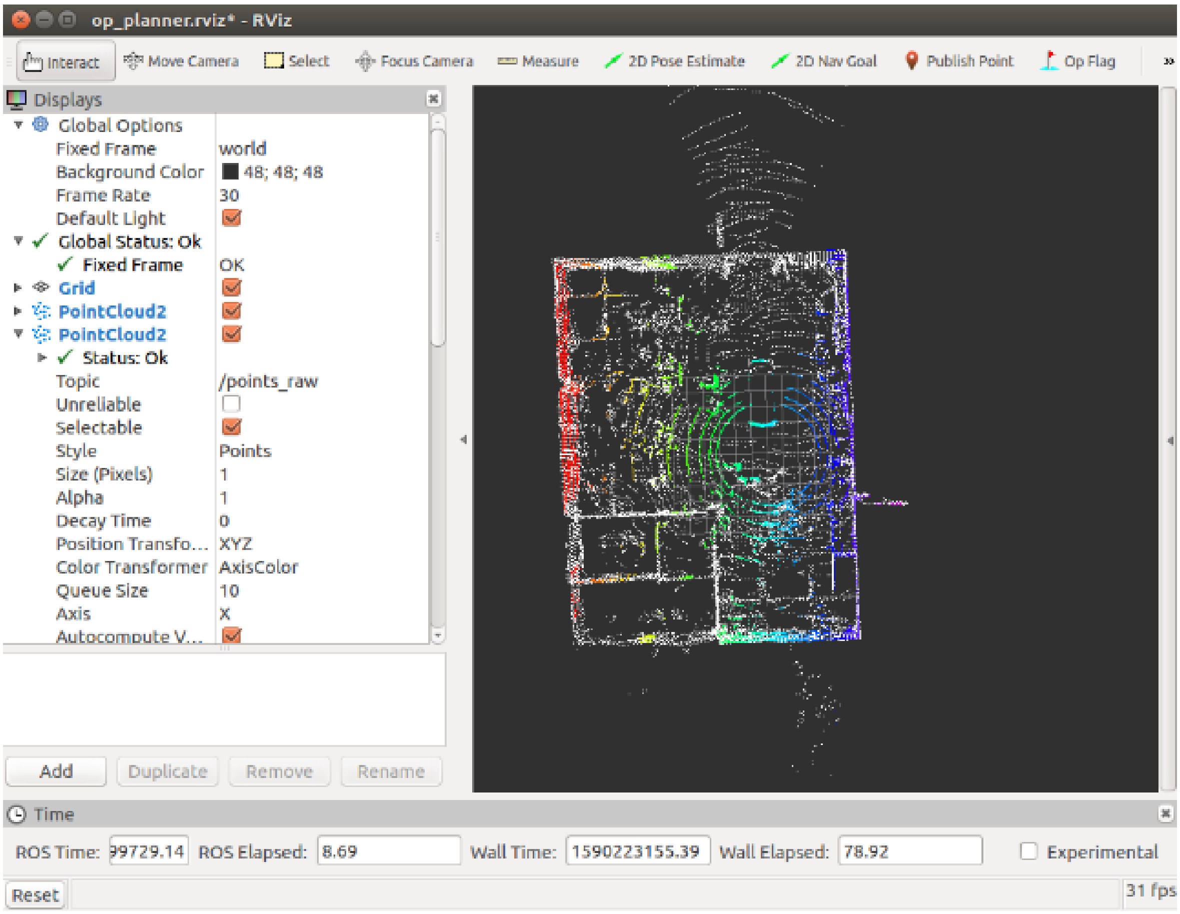
Start the speed conversion: Here to transform the /twist_amd subject of Autoware to the /cmd_vel subject that we can control

Th chassis need to be placed at the starting point location at this moment

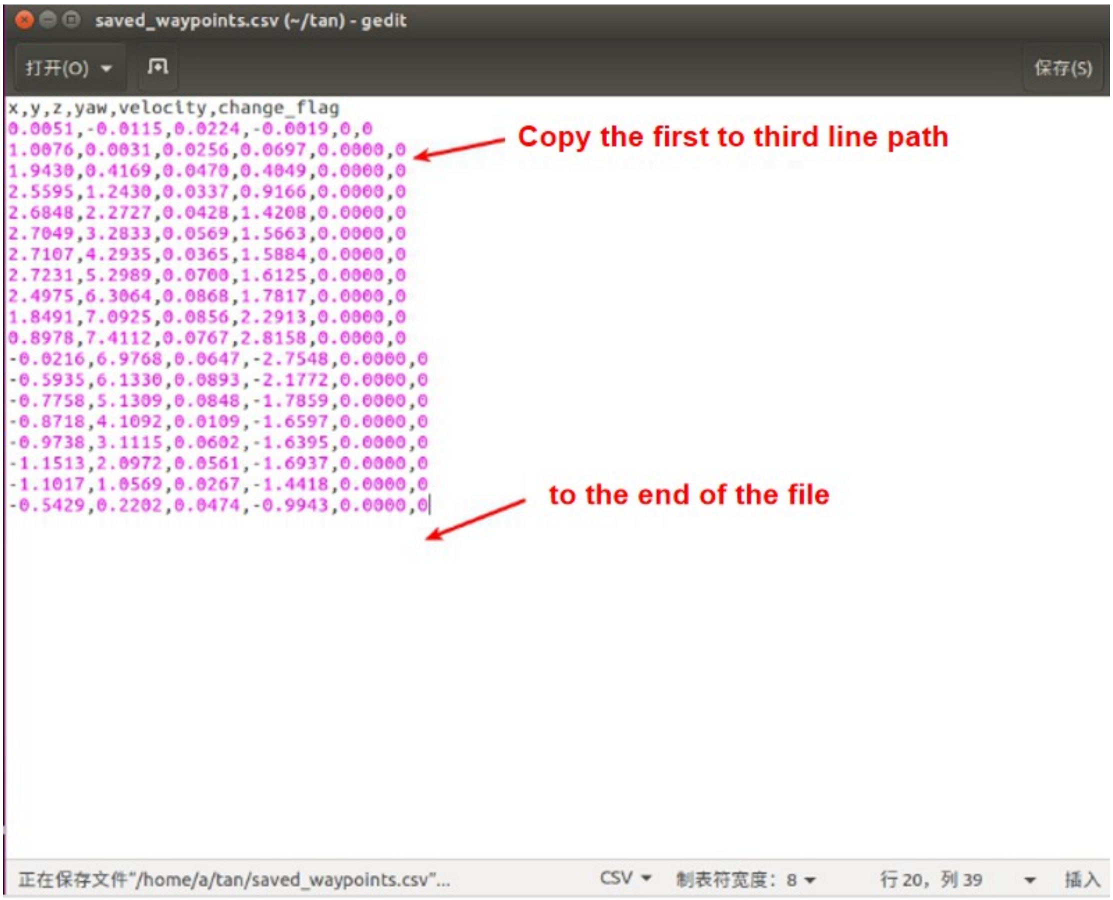
$ rosrun hunter_robot speed

Open the RVIZ, loading Autoware-1.9,0/rviz/op_planner.rviz file


Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Topics
State
nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys] [app]
ndt_matching [sys] [app]
ndt_matching_monitor [sys] [app]
icp_matching [sys] [app]
autoware_connector
can2odom [sys]
vel_pose_connect [sys] [app]
Detection
vision_detector
vision_dpm_ttic_detect [sys] [app]
vision_ssd_detect [sys] [app]
vision_darknet_yolo3 [sys] [app]
vision_darknet_yolo2 [sys] [app]
vision_tracker
vision_klt_track [sys] [app]
vision_kf_track [sys] [app]
vision_dummy_track [sys]
lidar_detector
astar_planner
obstacle_avoid [sys] [app]
velocity_set [sys] [app]
lattice_planner
lattice_velocity_set [sys] [app]
path_select [sys]
lattice_trajectory_gen [sys] [app]
lattice_twist_convert [sys]
waypoint_maker
waypoint_loader [sys] [app]
waypoint_saver [sys] [app]
waypoint_clicker [sys] [app]
waypoint_velocity_visualizer [sys] [app]
waypoint_follower
pure_pursuit [sys] [app]
twist_filter [sys] [app]
wf_simulator [sys] [app]
Decision
state_machine(old) [sys]
decision_maker [sys] [app]
planner_selector [sys] [app]
①Speed velocity
② Path select
③Saving path
④ Tracking
⑤Speed filter
Synchronization
ROSBAG
RViz
RQT
0.0%
0.0%
0.0%
0.0%
0.0%
0.0%
9.1%
0.0%
CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
top (18.2 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0H] (0.0 %CPU)
[mm_percpu_wq] (0.0 %CPU)
1GB/15GB(12%)
Memory
AutoWare



Additional content: If you want the chassis to drive in a circle, copy the saved_waypoints.csv file 1st to 3rd line path to the end.




```
x,y,z,yaw,velocity,change_flag
0.0051,-0.0115,0.0224,-0.0019,0,0
1.0076,0.0031,0.0256,0.0697,0.0000,0
1.9430,0.4169,0.0470,0.4049,0.0000,0
2.5595,1.2430,0.0337,0.9166,0.0000,0
2.6848,2.2727,0.0428,1.4208,0.0000,0
2.7049,3.2833,0.0569,1.5663,0.0000,0
2.7107,4.2935,0.0365,1.5884,0.0000,0
2.7231,5.2989,0.0700,1.6125,0.0000,0
2.4975,6.3064,0.0868,1.7817,0.0000,0
1.8491,7.0925,0.0856,2.2913,0.0000,0
0.8978,7.4112,0.0767,2.8158,0.0000,0
-0.0216,6.9768,0.0647,-2.7548,0.0000,0
-0.5935,6.1330,0.0893,-2.1772,0.0000,0
-0.7758,5.1309,0.0848,-1.7859,0.0000,0
-0.8718,4.1092,0.0109,-1.6597,0.0000,0
-0.9738,3.1115,0.0602,-1.6395,0.0000,0
-1.1513,2.0972,0.0561,-1.6937,0.0000,0
-1.1017,1.0569,0.0267,-1.4418,0.0000,0
-0.5429,0.2202,0.0474,-0.9943,0.0000,0
```

5) Use hybrid A* to autonomous navigation

Start Autoware:

```
$ cd Autoware-1.8.0/ros/
$ ./run
```

Start lidar:

```
$ roslaunch rslidar_pointcloud rs_lidar_16.launch
```

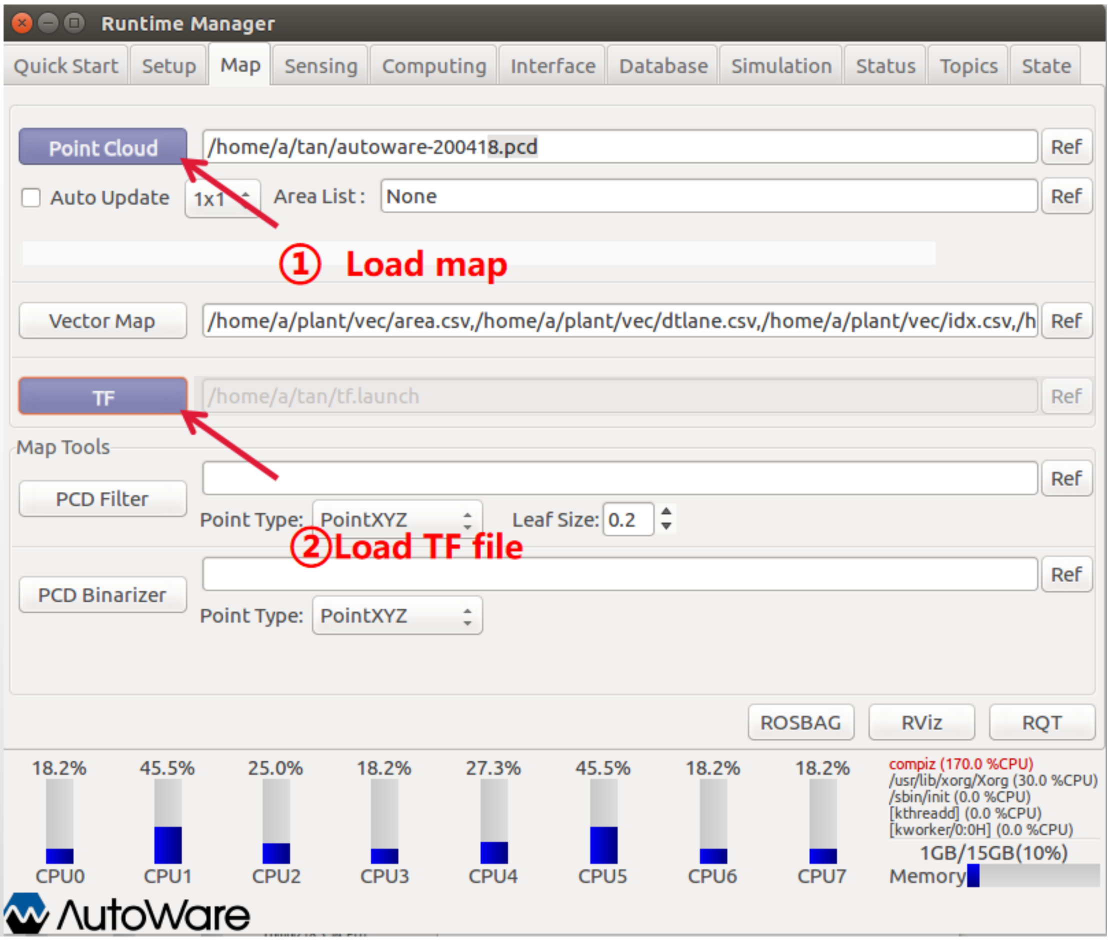
Start the chassis:

```
$ rosrun hunter_robot hunter_robot
```

Start speed transformation:

```
$ rosrun hunter_robot speed
```

Enter the map module of Autoware, select and load Point Cloud and TF



Enter Sensing module, loading vexel_grid_filter and Virtual Scan Image

Enter Computing module

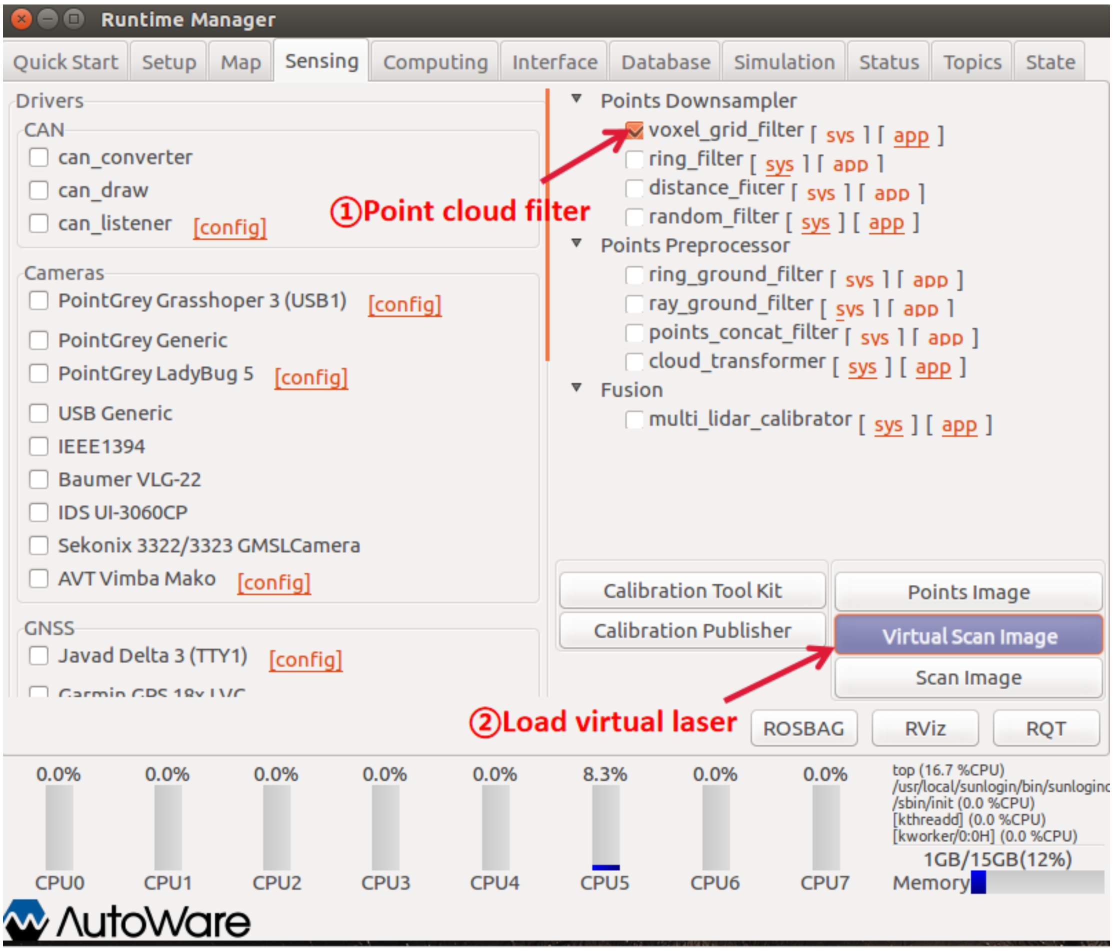
Click and loading Localization->lidar_localizer->ndt_matching

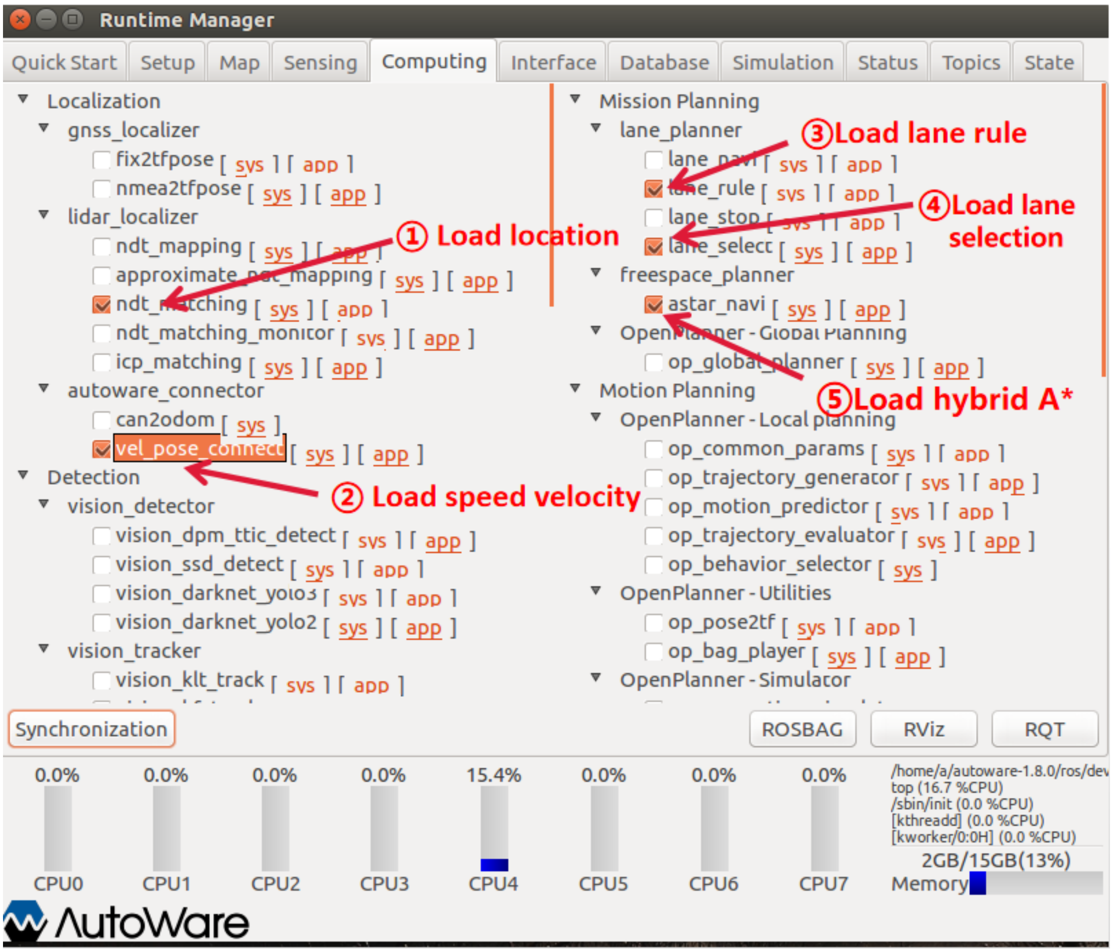
Localization->Autoware_connector->vel_pos_connect

Mission Planning->lane_planner->lane_rule

Mission Planning->lane_planner->lane_rule

Mission Planning->freespace_planner->astar_navi


Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Topics
State
Drivers
CAN
can_converter
can_draw
can_listener [config]
Cameras
PointGrey Grasshoper 3 (USB1) [config]
PointGrey Generic
PointGrey LadyBug 5 [config]
USB Generic
IEEE1394
Baumer VLG-22
IDS UI-3060CP
Sekonix 3322/3323 GMSLCamera
AVT Vimba Mako [config]
GNSS
Javad Delta 3 (TTY1) [config]
①Point cloud filter
Points Downsampler
voxel_grid_filter [sys] [app]
ring_filter [sys] [app]
distance_filter [sys] [app]
random_filter [sys] [app]
Points Preprocessor
ring_ground_filter [sys] [app]
ray_ground_filter [sys] [app]
points_concat_filter [sys] [app]
cloud_transformer [sys] [app]
Fusion
multi_lidar_calibrator [sys] [app]
Calibration Tool Kit
Calibration Publisher
Points Image
Virtual Scan Image
Scan Image
②Load virtual laser
ROSBAG
RViz
RQT
0.0%
0.0%
0.0%
0.0%
0.0%
8.3%
0.0%
0.0%
CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
top (16.7 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0H] (0.0 %CPU)
1GB/15GB(12%)
Memory
AutoWare



Semantics->laserscan2costmap

Motion_planning->astar_planner->obstacle_avoid

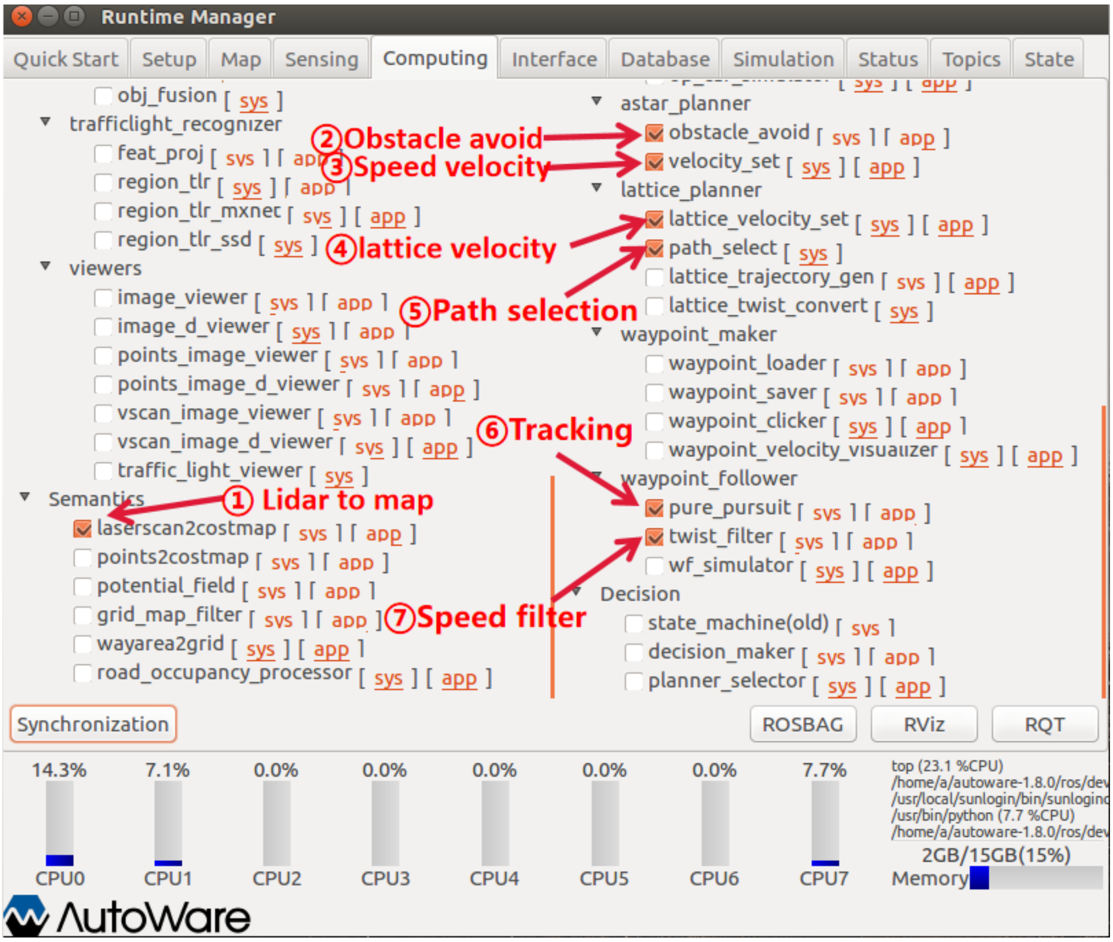
Motion_planning->astar_planner->velocuty_set

Motion_planning->lattice_planner->lattice_velocity_set

Motion_planning->lattice_planner->path_select

Motion_planning->waypoint_follower->pure_pursuit

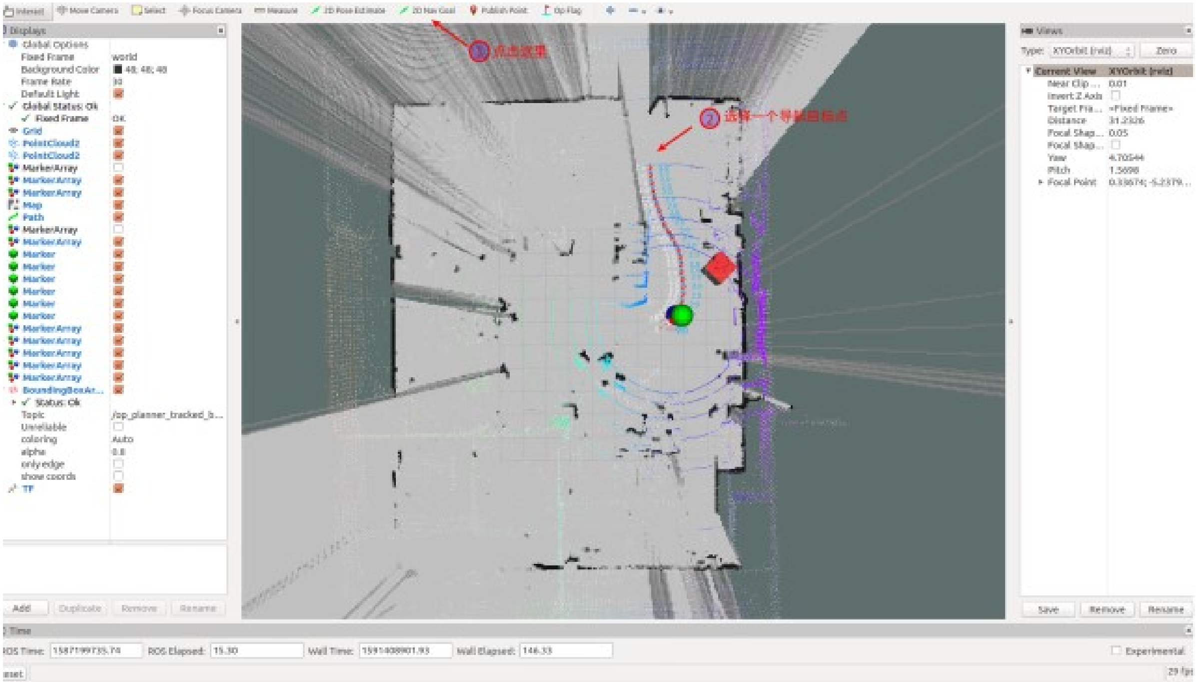
Motion_planning->waypoint_follower->twist_filter



Open RVIZ, loading the Autoware-1.8.0/rviz/op_planner.rviz file

Then select a navigation target point on RVIZ, you can see that a navigation path is generated, and the car follows this path to the navigation target point.

Note: Obstacles cannot be avoided during the planning process, it is needed to scan by lidar before the planning to avoid the obstacles.



6) Use Autoware for local path planning

Local planning has to combined with tracking along the line or global planning. Here is the instruction for how to track along the line.

Start Autoware:

$ cd Autoware-1.8.0/ros/

$ ./run

Start lidar:

$ roslaunch rslidar_pointcloud rs_lidar_16.launch

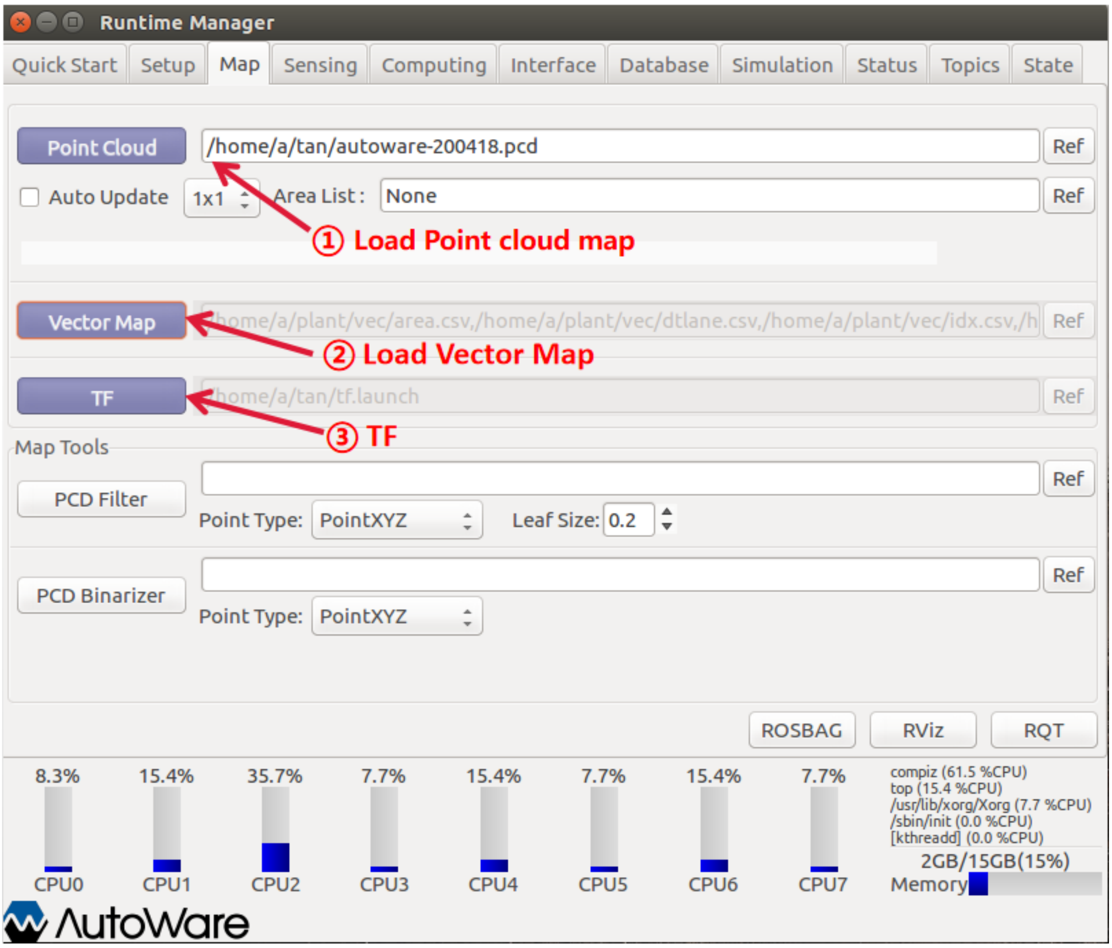
Start chassis:

$ rosrun hunter_robot hunter_robot

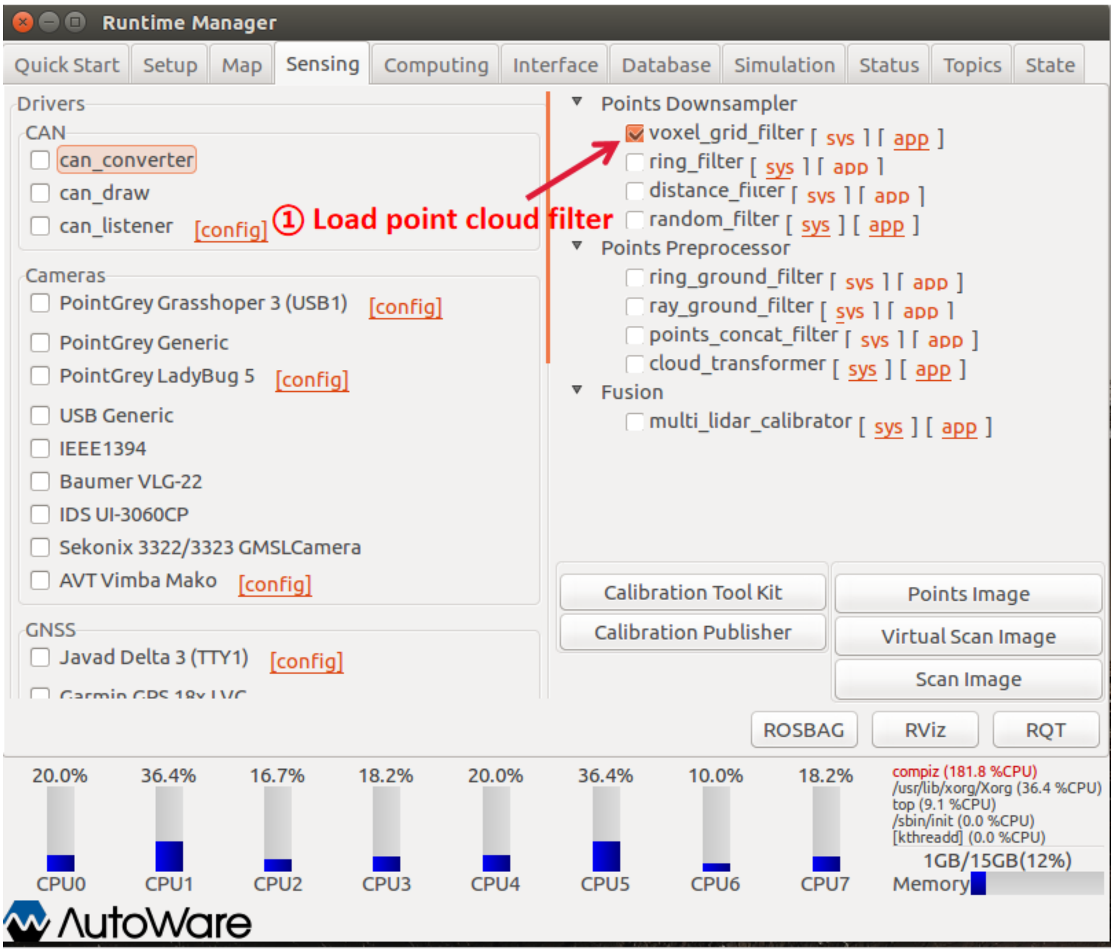
Start speed conversion:

$ rosrun hunter_robot speed

Enter the map module of Autoware, select and loading Point Cloud and TF



Enter Sensing module, downloading Points Downsampler->vexel_grid_filter



Enter Computing module, loading

Localization->lidar_localizer->ndt_matching

Localization->Autoware_connector->vel_pose_connect

Mission_Planning→lane_planner->lane_rule

Mission_Planning→lane_planner→lane_select

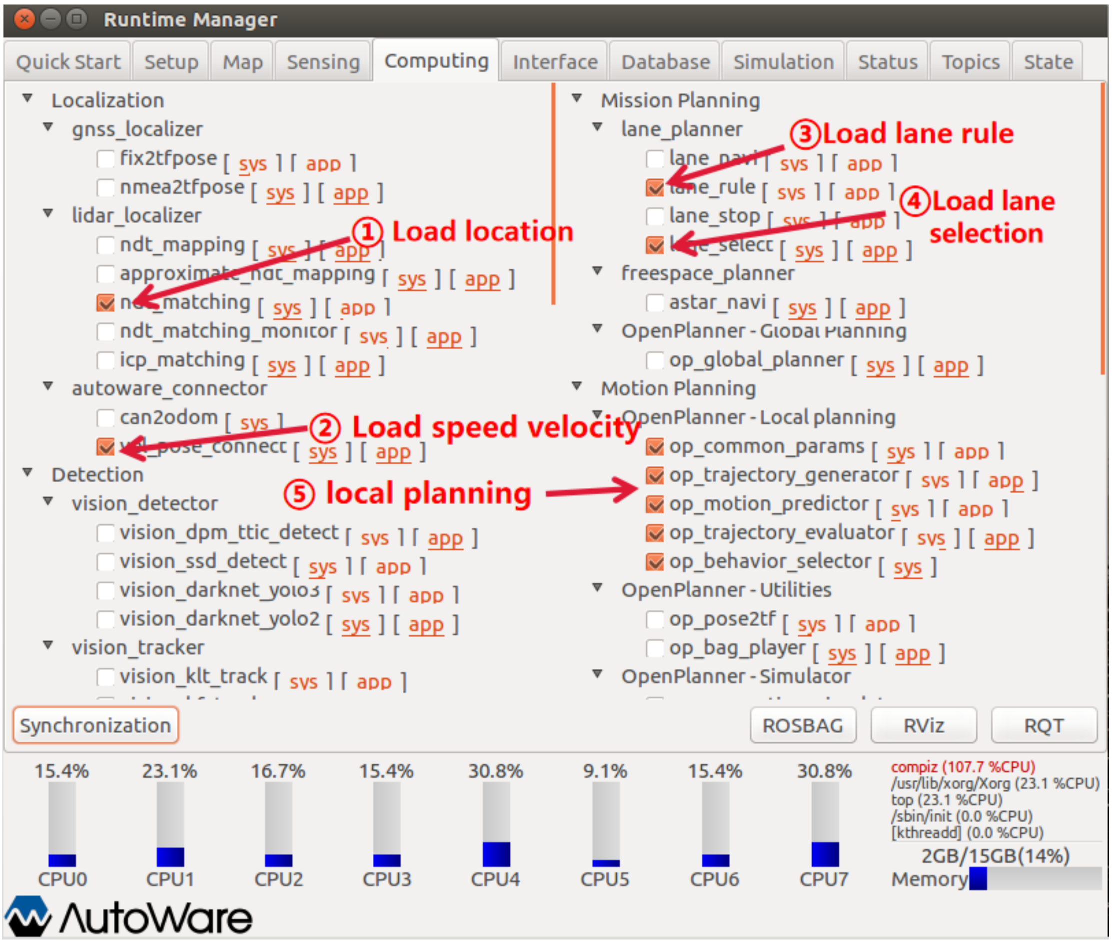
Mossion_planning→OpenPlanner-Local planning→op_common_params

Mossion_planning→OpenPlanner-Local planning→op_trajectory_generator

Mossion_planning→OpenPlanner-Local planning→op_motion_predictor

Mossion_planning→OpenPlanner-Local planning→op_trajectory_evaluator

Mossion_planning→OpenPlanner-Local planning->op_behavior_selector



Loading

Detection→lidar_tracker→lidar_kf_contour_track

Motion Planning->OpenPlanner-Simulator->op_perception_simulator

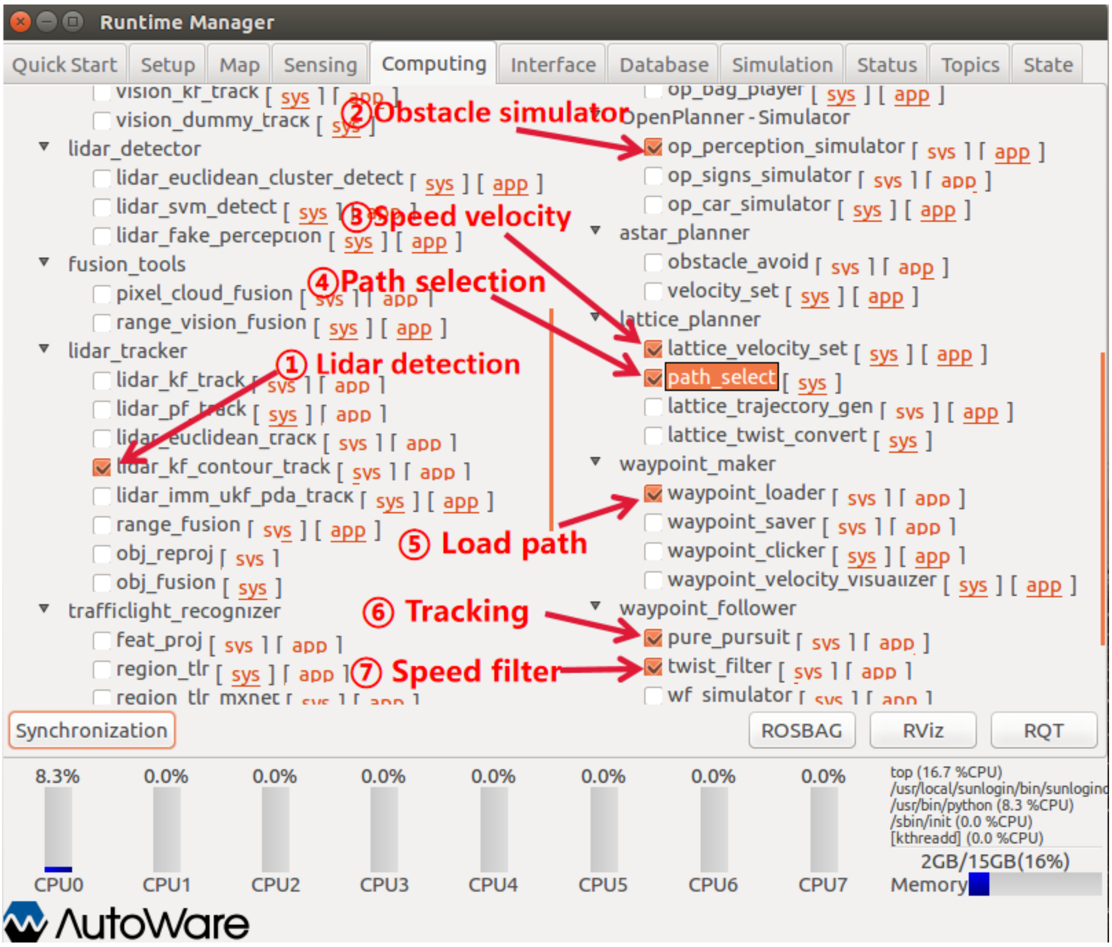
Motion Planning->lattice_planner->lattice_velocity_set

Motion Planning→lattice_planner→path_select

Motion Planning→waypoint_marker→waypoint_loader

Motion Planning->waypoint_follower->pure_pursuit

Motion Planning->waypoint_follower->twist_filter

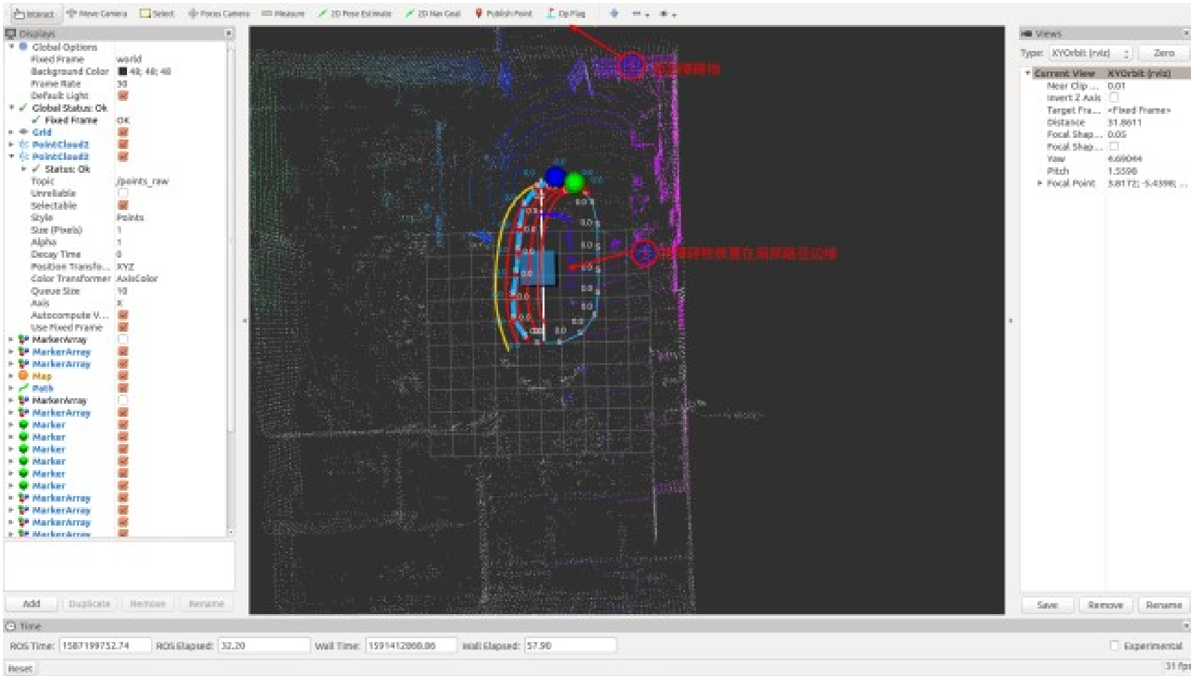


Open RVIZ, download Autoware-1.8.0/rviz/op_planner.rviz file

You can see that the car is walking along a fixed line, and there are also several partially planned paths around.

At this moment, a virtual obstacle can be added and placed at the edge of the local path, Autoware is planning to pass this virtual obstacle.

If you need to avoid obstacles, you need to add camera recognition or point cloud clustering (the function has been improved)



7) Use Autoware for global planning

Combined with local planning, simulation for obstacles passing

Start Autoware:

```
$ cd Autoware-1.8.0/ros/
$ ./run
```

Start the lidar:

```
$ roslaunch rslidar_pointcloud rs_lidar_16.launch
```

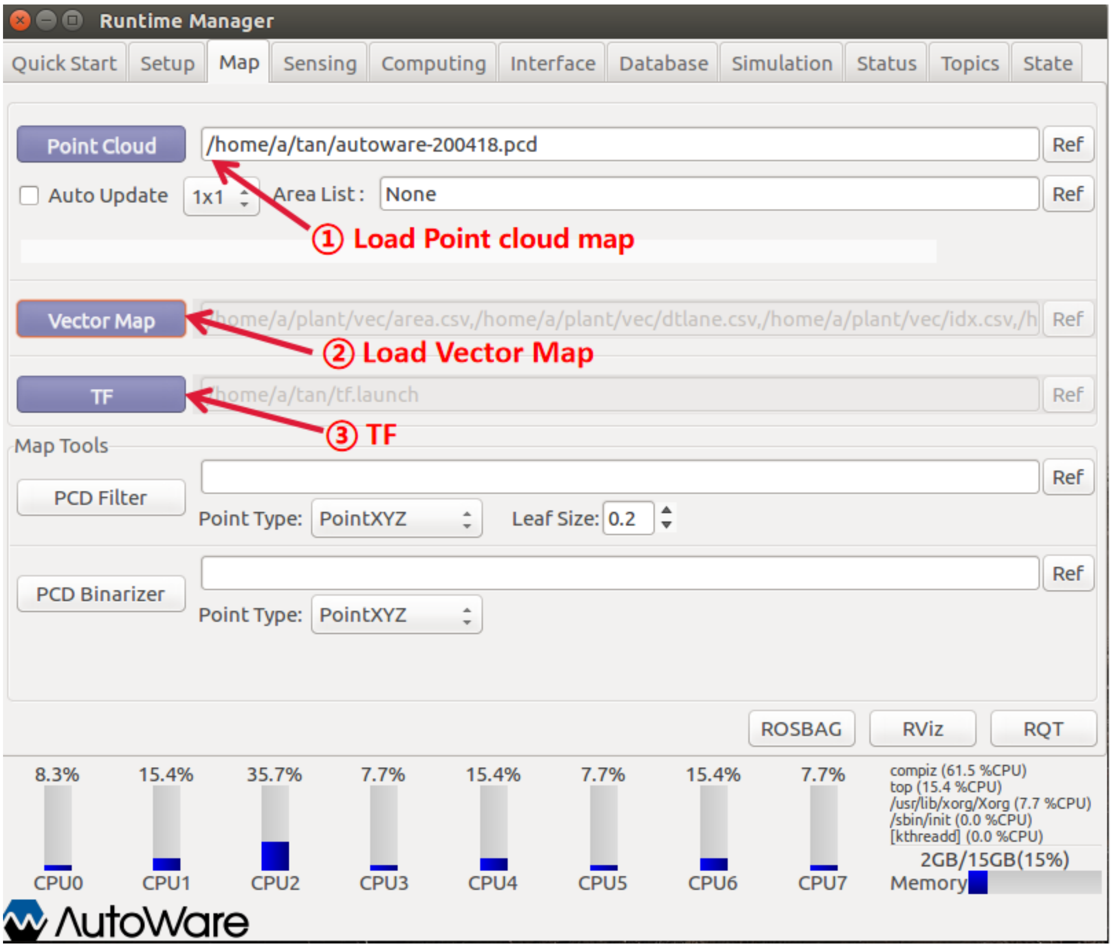
Start the chassis:

```
$ rosrun hunter_robot hunter_robot
```

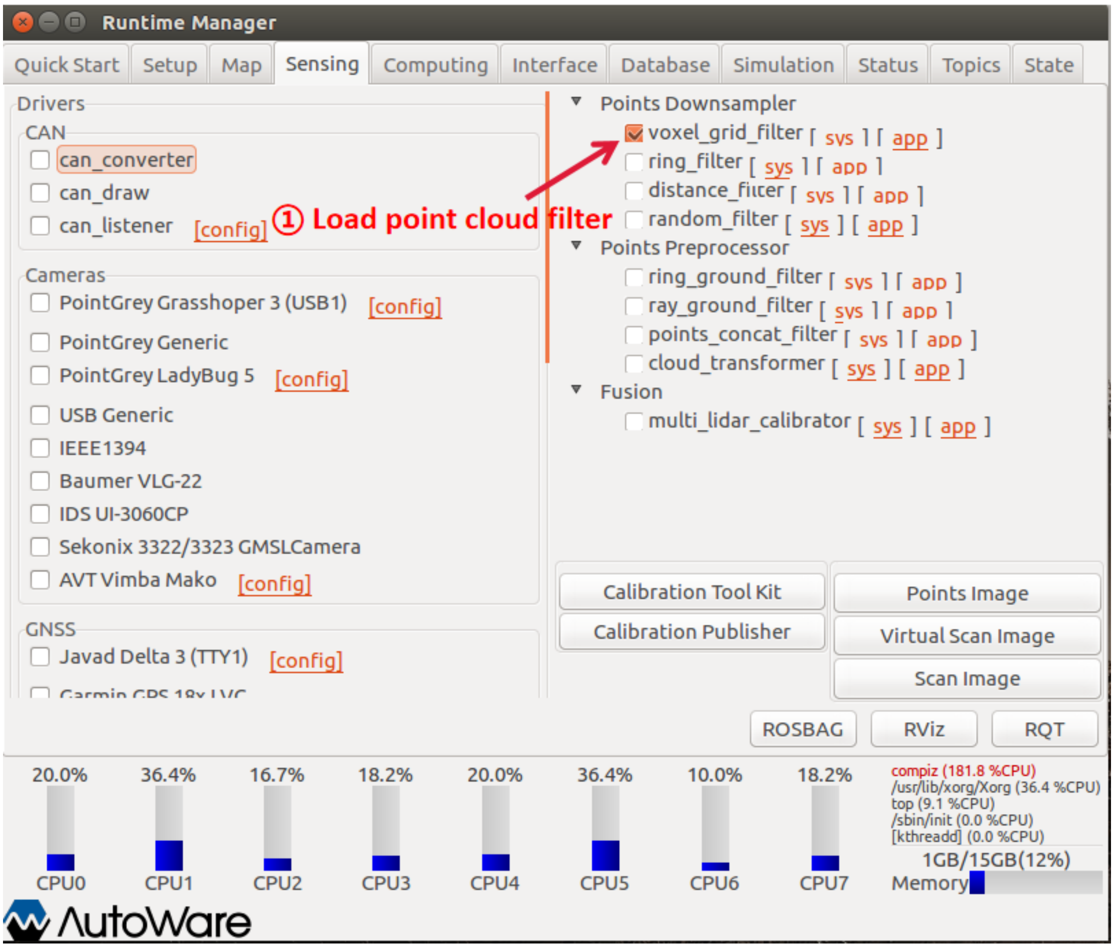
Start speed transformation:

```
$ rosrun hunter_robot speed
```

Enter the map module of Autoware, select and load Point Cloud, Vector Map and TF



Enter Sensing module, load Points Downsampler->vexel_grid_filter



Enter Computing module and loading

Localization->lidar_localizer->ndt_matching

Localization->Autoware_connector->vel_pose_connect

Mission_Planning→lane_planner->lane_rule

Mission_Planning→lane_planner→lane_select

Mission_Planning→OpenPlanner-Global Planning->op_global_planner

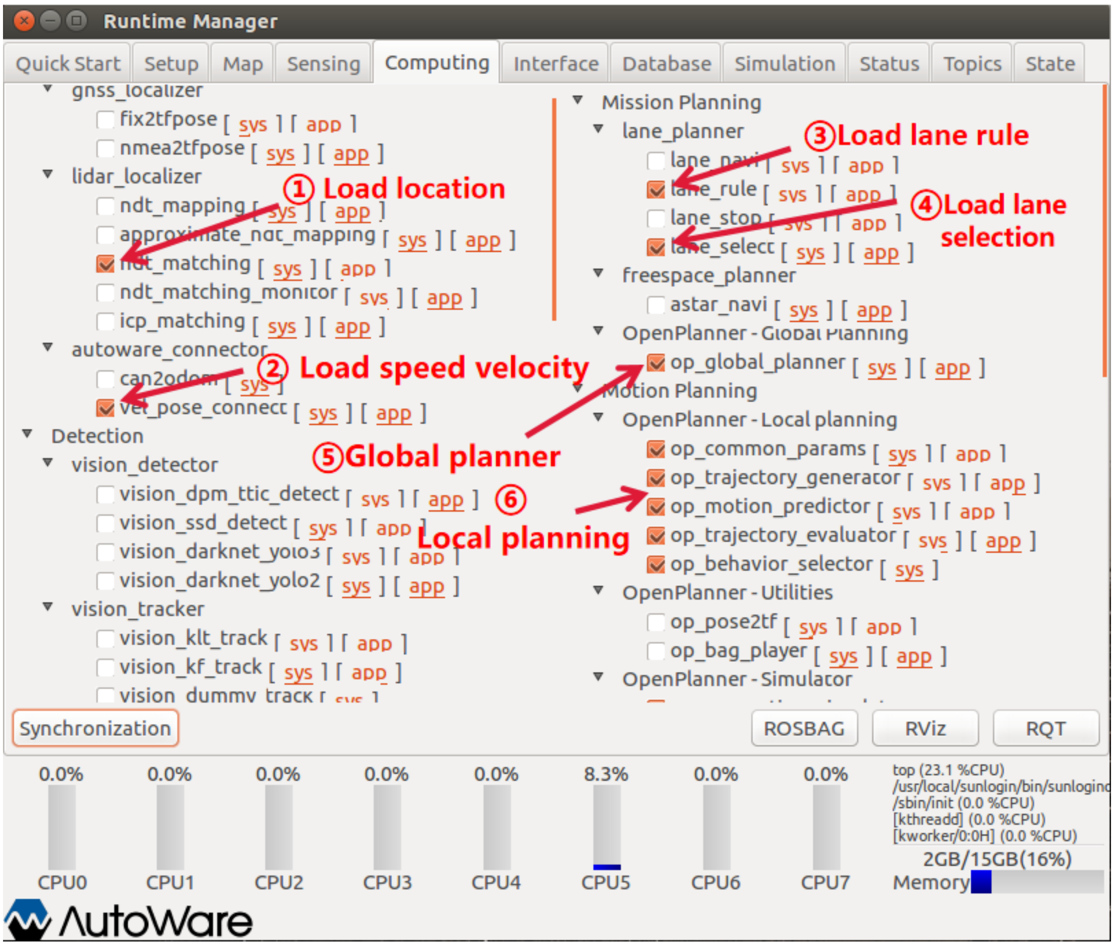
Mossion_planning→OpenPlanner-Local planning→op_common_params

Mossion_planning→OpenPlanner-Local planning→op_trajectory_generator

Mossion_planning→OpenPlanner-Local planning→op_motion_predictor

Mossion_planning→OpenPlanner-Local planning→op_trajectory_evaluator

Mossion_planning→OpenPlanner-Local planning->op_behavior_selector



Loading

Detection→lidar_tracker→lidar_kf_contour_track

Motion Planning->OpenPlanner-Simulator->op_perception_simulator

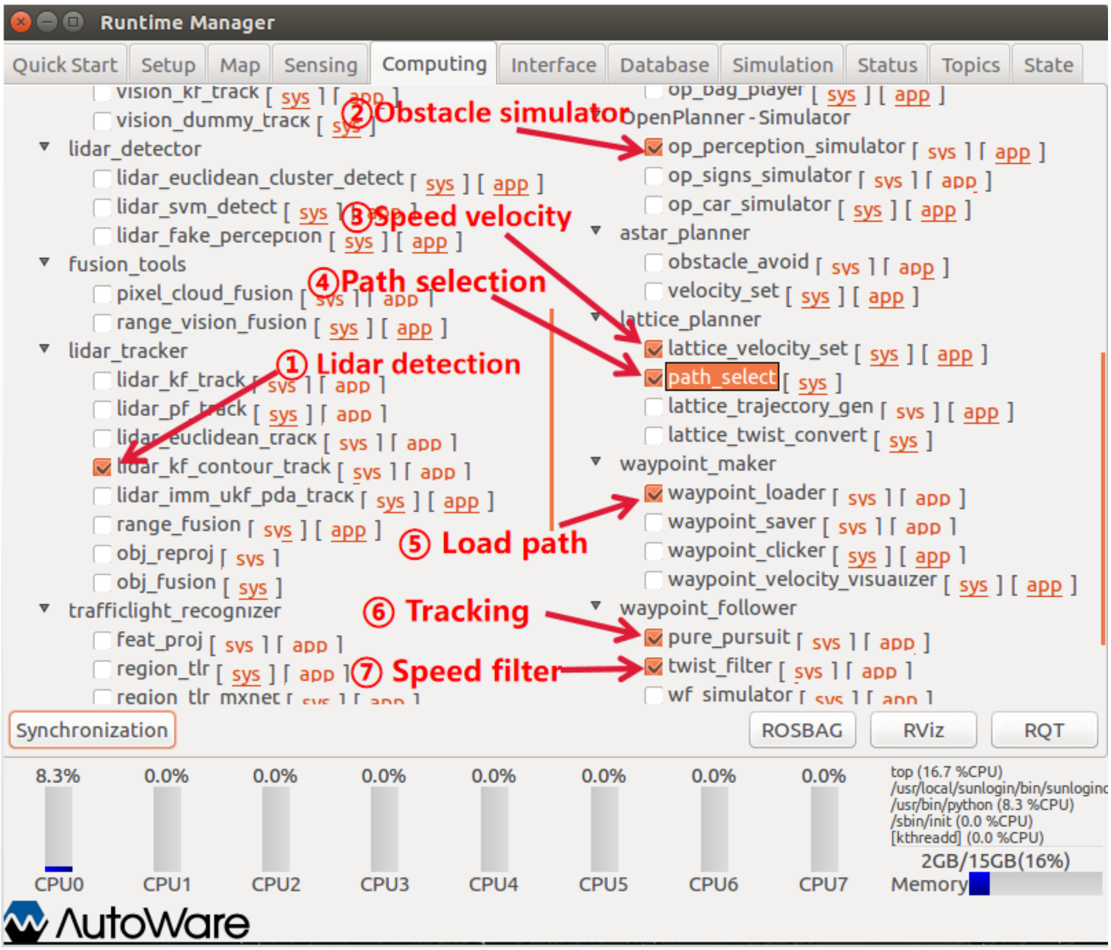
Motion Planning->lattice_planner->lattice_velocity_set

Motion Planning→lattice_planner→path_select

Motion Planning→waypoint_marker→waypoint_loader

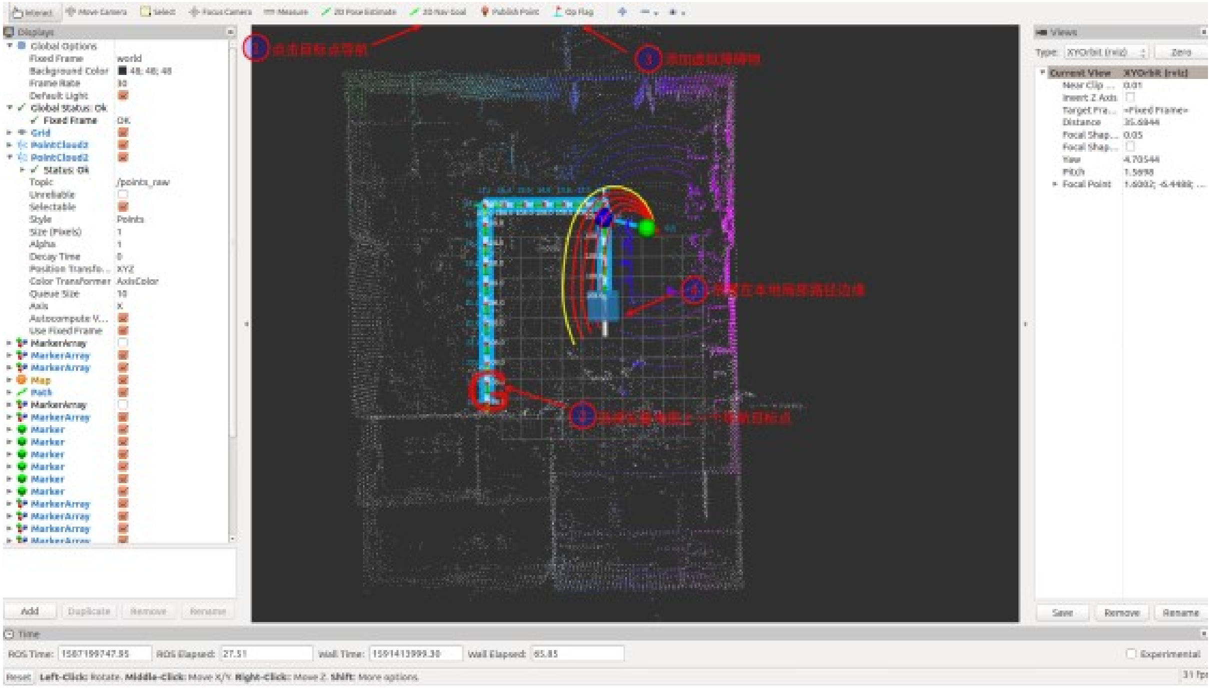
Motion Planning->waypoint_follower->pure_pursuit

Motion Planning->waypoint_follower->twist_filter



Open RVIZ, loading sutoware-1.8.0/rviz/op_planner.rviz file

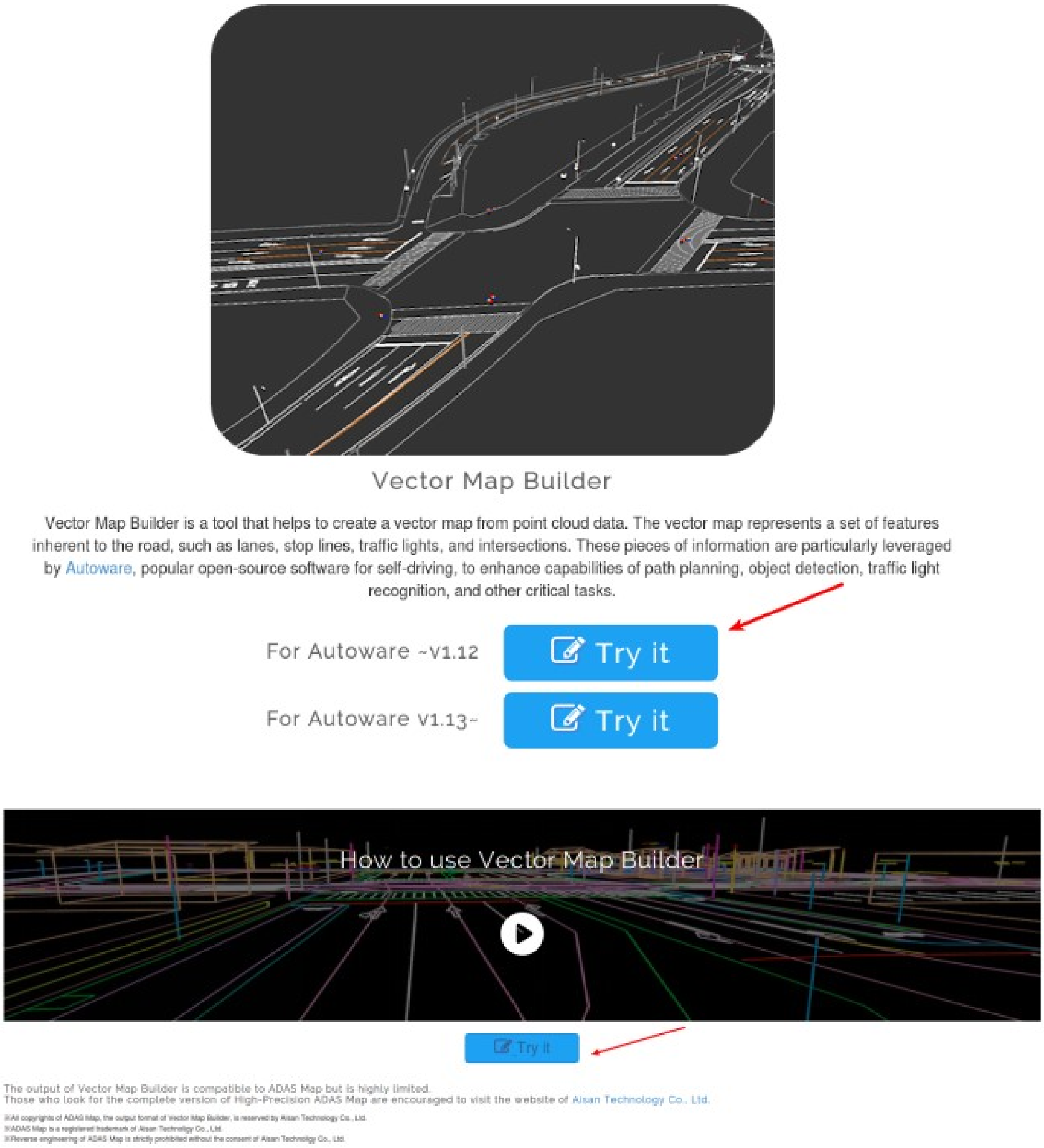
Move the mouse to RVIZ, select a target point on the vector map, it would generate global planning path. Then adding virtual obstacles to the edge of the local path, and there would be a detour path.



4. Products and services

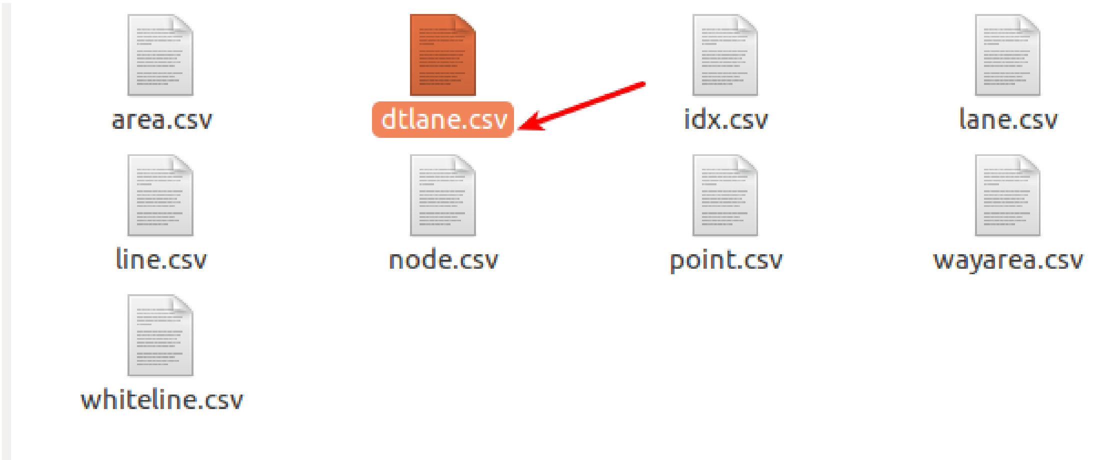
8) Edit vector map

Official website: https://tools.tier4.jp/
It is needed to edit the map on the website, register a tier account, log in and select the edit map page.



See the video for specific editing map tutorials

After exporting the map, it is needed to copy the dtlane.csv file manually to the .csv map file which just exported, and download the vector map later.



i) After-sales policy

a. Maintenance service

1. If the product is sold within 1 year(from the date of product acceptance and inspection, if the corresponding acceptance check receipt is not provided, the sold time would subject to 15 working days after signing the contract), if there is a any problems about the performance of product (No artificial cause damage), our company provides after-sales and maintenance services which based on the circumstances . If the product is sold more than 1 year (from the date of product acceptance and inspection, if the corresponding acceptance check receipt is not provided, the sold time would subject to 15 working days after signing the contract), then lifetime maintenance paid service is available.

2. Customers need to pay for the freight in the following situations
   1) The situation is not covered by the warranty.
   2) Product return for after-sales service application.
   3) Product inspection does not meet the conditions of return and exchange terms.

b. Warranty

1. You can enjoy free maintenance service when the product is sold with the following situations:
   1) The production does not work normally for the first time;
   2) If there is any problems about performances with proper operation. (No artificial cause damage);
   3) Free for maintenance or accessories replacing if any problems or damages occur within 15 days from the date of inspection checking;
   4) The following five warranty accessories would not be maintenance for free if there is any problems after the acceptance completed and accepted.

c. Non-warranty coverage accessories: Warranty time period table

HUNTER	Tyre	No warranty
	Appearance sheet metal parts	No warranty
	Power system(Motor/ Timing belt/ Reduction box/ Cardan joint)	6 months
	Power battery	6 months
	Charger	12 months
	Main control panel	12 months
	Motor drive board	12 months
	Remote control	12 months
	Remote control receiver	12 months
Lidar		12 months
IPC		12 months
4G router		12 months
LCD		12 months

d. Non-warranty coverage

Warranty service is unavailable in the following situations :

1. Artificial damage, including unauthorized dismantling of the machine, collision and so on ;
2. The relevant proof of purchase is not provided, or the proof of purchase content does not match the product;
3. The proof of the purchase content has been altered or blurred and cannot be identified;
4. Force majeure.

e. Paid technical support service available in the following situations:

1. Fail to follow the manual instruction to operate the machine and force majeure
2. Artificial cause damage, such as falling, squeezing, immersion, etc.;
3. The machine have been repaired by other companies;
4. Changing or using the other company’s accessories to cause machine breakdown or damage;
5. Other damages not caused by products or accessories;
6. Place the product in a condition that exceeds its own environmental limitation: such as corrosion, oxidation,burns and excessive humidity cause by environmental rapid changing;

7. Proof Product purchase and sales company is not provided;
8. The purchase date exceed the warranty coverage period.

ii) Technical support service

1) This product provides the concept of education development, let everyone enjoys the fun of self-driving . As the initiator of Autoware, we are willing to work together to discuss the solution and solving problems.
2) Provide limited technical support and development guidance service.

iii) Value added service

1) Off-line training service (Basic training for existing product) ¥3000 one person per day

5. Frequently asked question

6. Development advice and guidance