

# Introduction of Racecar

主讲人：

超杰

# 目录

1

比赛及框架介绍

2

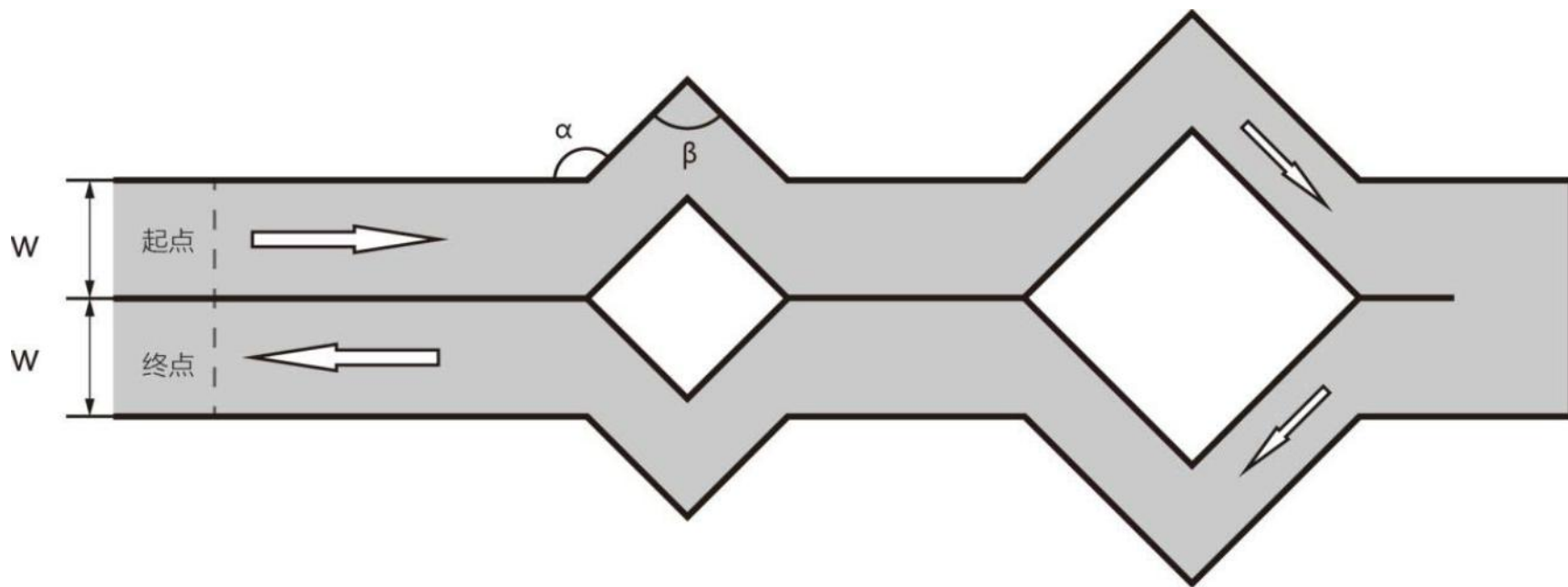
传感器使用

3

软件层参数配置

### □ 本次室外无人驾驶创意赛赛道分析：

赛道总长度在50m~100m 之间，赛道宽度在2m~3m 之间，赛道是由横幅或其他材质围挡起来，围挡的赛道高度在30cm~50cm。赛道由多处折弯，其中赛道 $\alpha$ 角的范围在 $120^\circ\sim 160^\circ$ ， $\beta$ 角的范围在 $90^\circ\sim 130^\circ$ ，具体如图所示：





□ 演示视频:



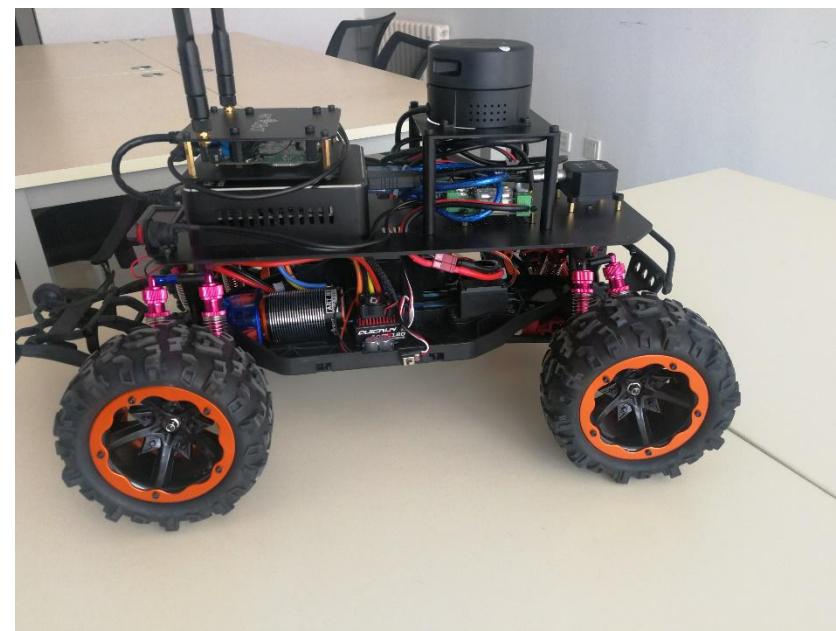
<https://v.qq.com/x/page/v08085vx8ta.html>

# Why Racecar

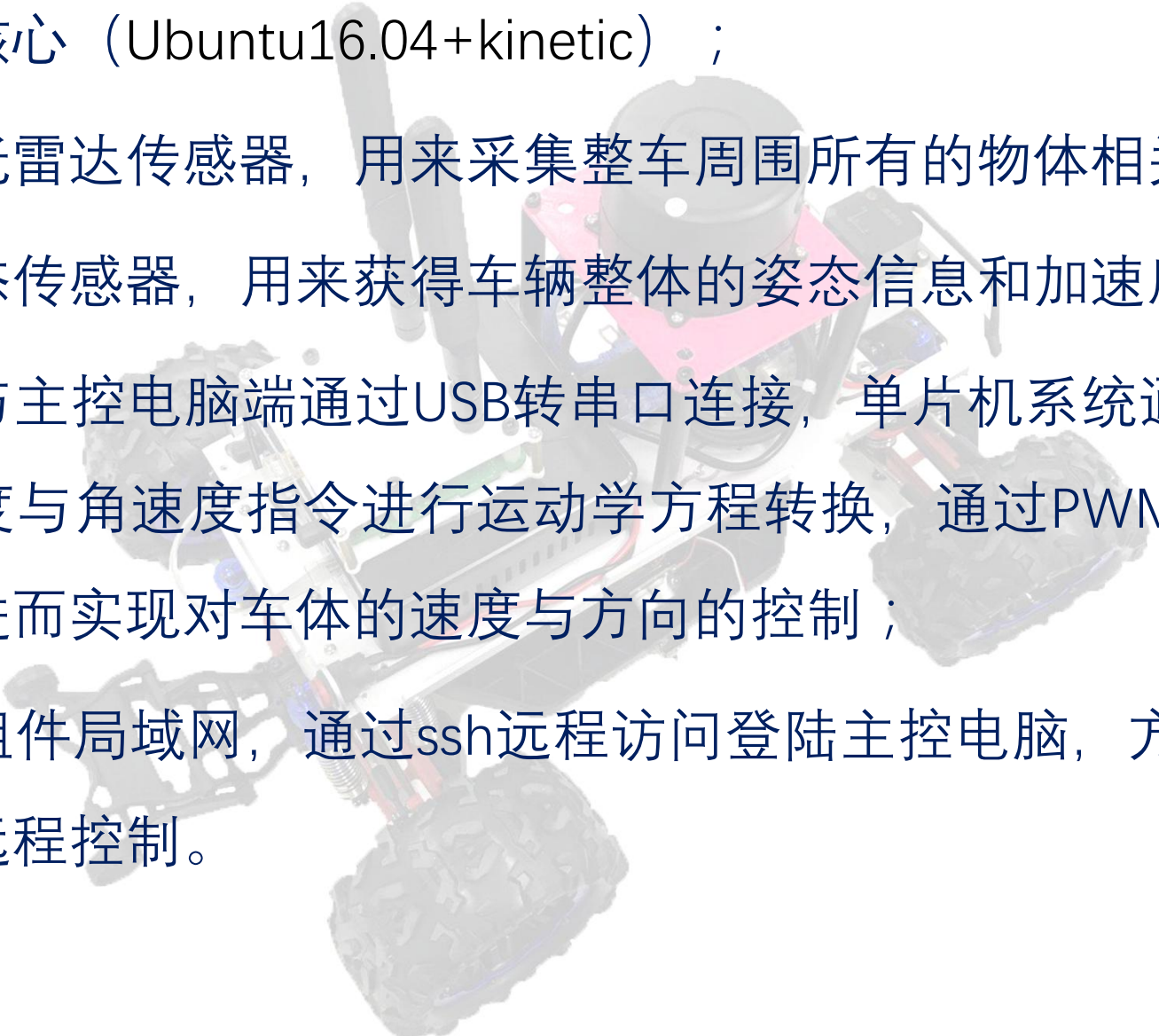
- ◆ ROS开源支持
- ◆ 能够实现二维导航（基于激光）
- ◆ 高精度（恩智浦、民用级激光雷达、IMU等）
- ◆ 高性能
- ◆ 算法评估及比较
- ◆ 模块化以及可扩展性



## 硬件平台





- 
- 主控电脑为核心（Ubuntu16.04+kinetic）；
  - 外部搭载激光雷达传感器，用来采集整车周围所有的物体相关的距离信息；
  - 同时搭载姿态传感器，用来获得车辆整体的姿态信息和加速度信息；
  - 单片机系统与主控电脑端通过USB转串口连接，单片机系统通过接收ROS系统下达的线速度与角速度指令进行运动学方程转换，通过PWM控制直流无刷电机和舵机，进而实现对车体的速度与方向的控制；
  - 通过路由器组件局域网，通过ssh远程访问登陆主控电脑，方便操作人员实现对整个系统远程控制。



IMU

雷达



打开Terminal，键入以下命令完成Catkin Workspace（默认创建功能包为imu\_t）初始化：

```
$ cd ~
```

```
$ mkdir -p ~/imu_t/src
```

之后将源代码包中imu部分代码移动到~/imu\_t/src目录，进一步执行：

```
$ cd ~/imu_t
```

```
$ catkin_make
```

编译完毕后键入

```
$ echo "source ~/imu_t/devel/setup.bash" >> ~/.bashrc
```

```
$ source ~/.bashrc
```

即完成了驱动安装。

在Rviz中显示IMU数据：

在Rviz中如果要想显示IMU数据的话，需要提前安装一个Rviz显示插件：

1>

```
$ sudo apt-get install ros-kinetic-imu-tools
```

2>

```
$ sudo apt-get install git-core
```

```
$ git clone -b <distro> https://github.com/ccny-ros-pkg/imu_tools.git
```

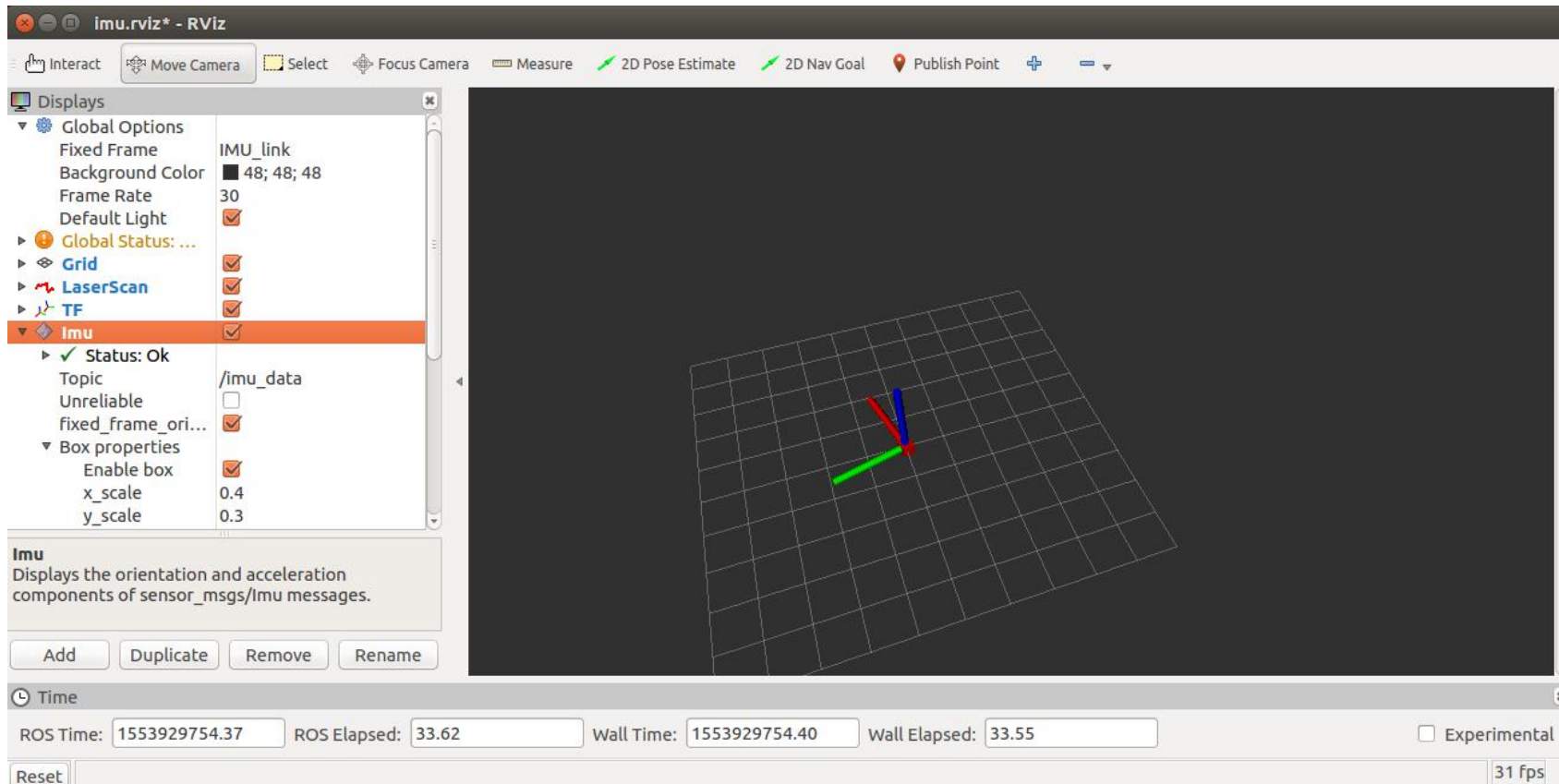
安装依赖：

```
$ rosdep install imu_tools
```

启动rviz可视化界面，如图：

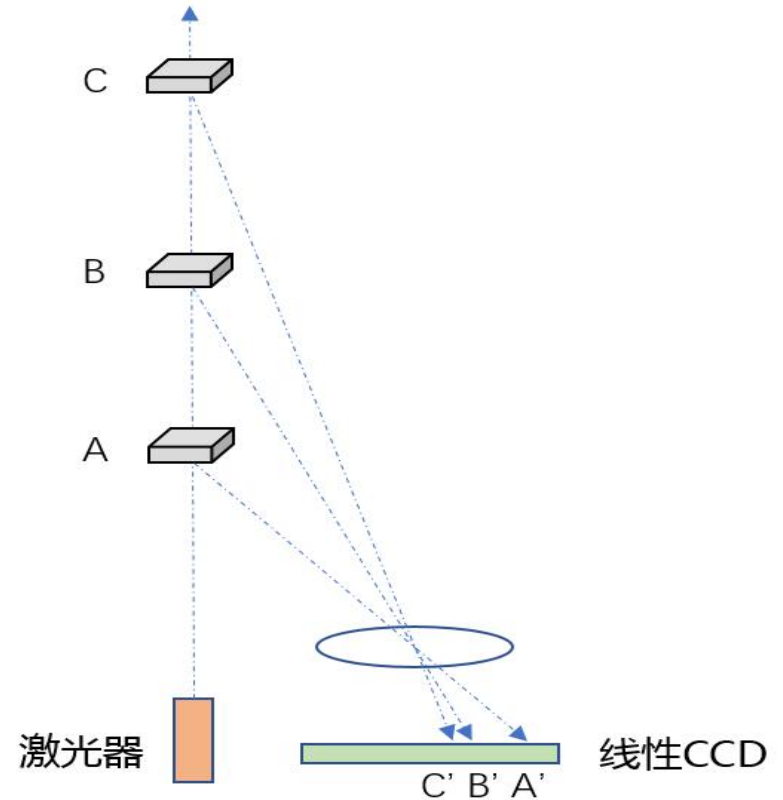
```
$ roslaunch art_imu imu_test.launch
```

```
$ rosrn rviz rviz
```



激光雷达的原理：

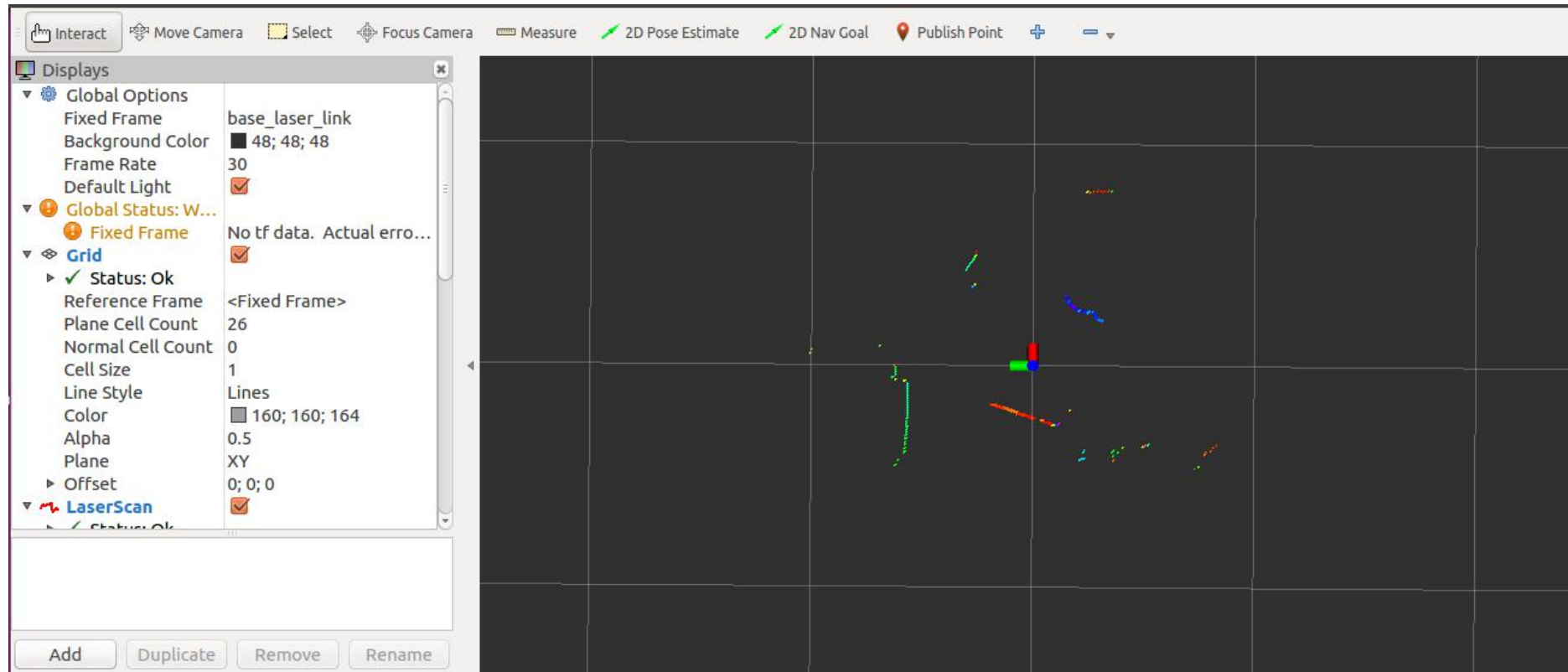
三角测距是一借由测量目标点与固定基准线的已知端点的角度，测量目标距离的方法。而不是直接测量特定位置的距离（三边量测法）。当已知一个边长及两个观测角度时，观测目标点可以被标定为一个三角形的第三个点。三角法的原理如图所示。



启动雷达，命令如下：

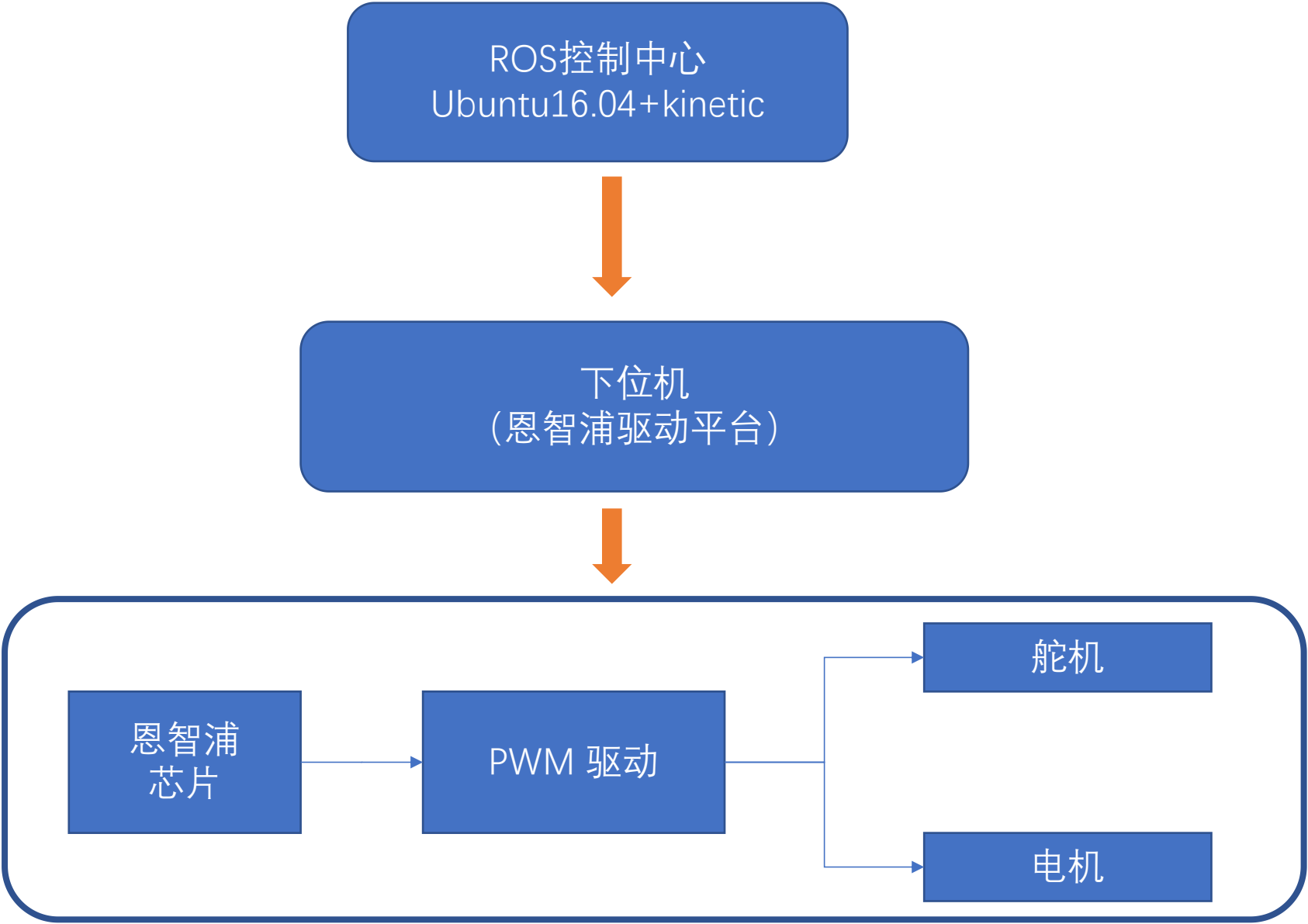
```
ubuntu@ubuntu:~$ roslaunch ls01g ls01g.launch
```

```
ubuntu@ubuntu:~$ roslaunch ls01g rviz.launch
```



RF2O是一种快速而精确的方法，用于从连续范围扫描中估计激光雷达的平面运动。对于每个扫描点，可以根据传感器速度制定范围流约束方程，并通过最小化所得几何约束的鲁棒函数以获得运动估计。

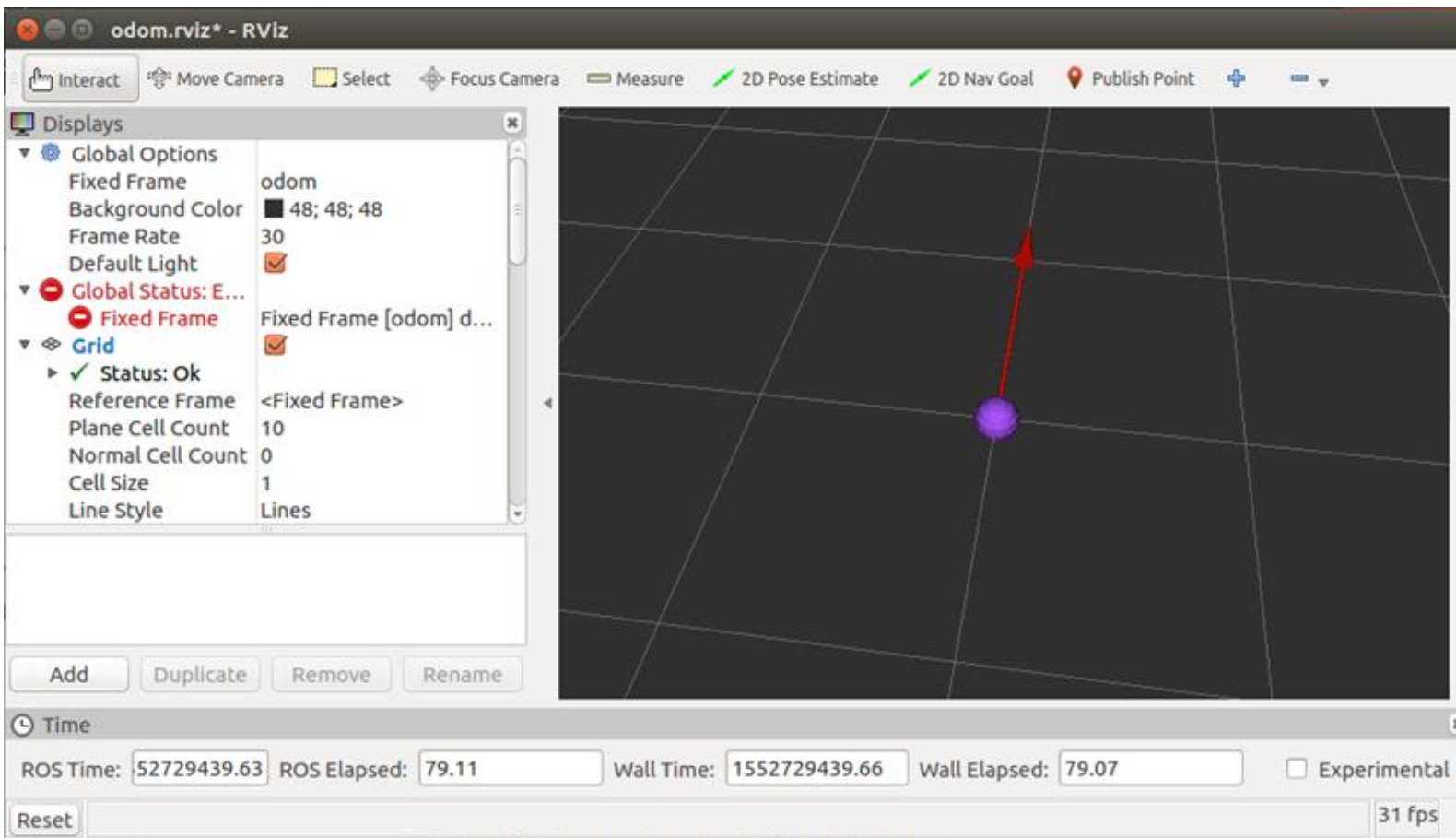
与传统方法相反，该方法不搜索对应关系，而是以密集的3D视觉测距法的方式，基于扫描梯度执行密集扫描对准。最小化问题以粗到精方案解决以应对大位移，并且基于估计的协方差的平滑滤波器用于处理无约束情景（例如走廊）中的不确定性。

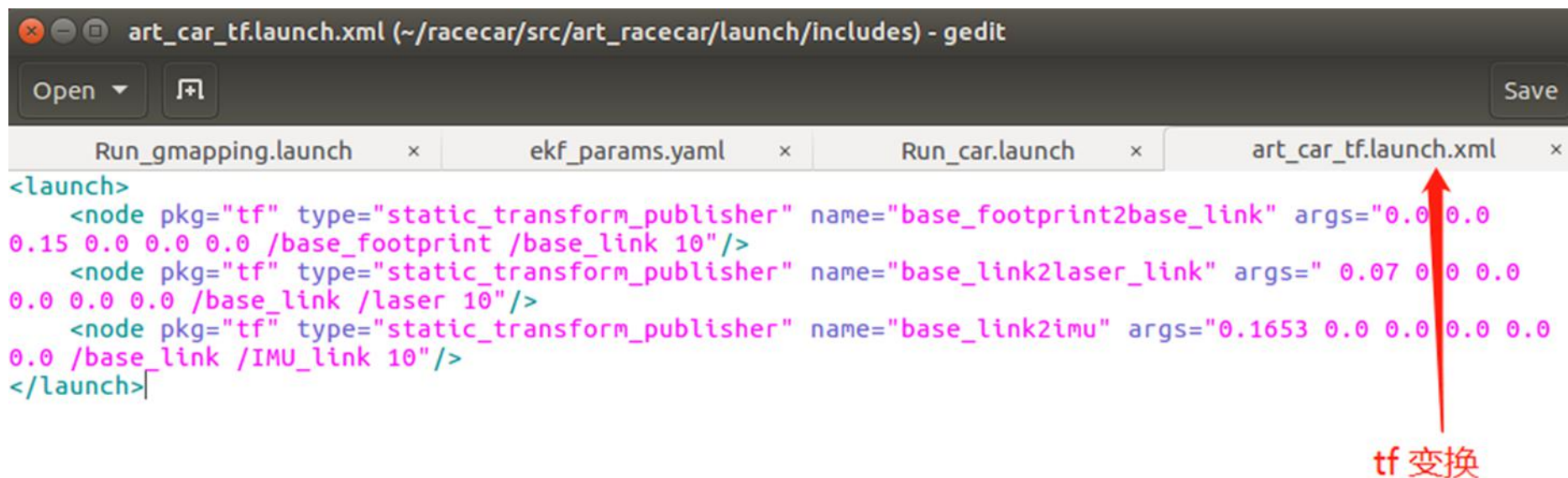




启动laser\_odom.launch文件进行可视化显示，操作如下：

```
$ roslaunch art_racecar laser_odom.launch
```





```
<launch>
  <node pkg="tf" type="static_transform_publisher" name="base_footprint2base_link" args="0.0 0.0
0.15 0.0 0.0 0.0 /base_footprint /base_link 10"/>
  <node pkg="tf" type="static_transform_publisher" name="base_link2laser_link" args=" 0.07 0 0 0.0
0.0 0.0 0.0 /base_link /laser 10"/>
  <node pkg="tf" type="static_transform_publisher" name="base_link2imu" args="0.1653 0.0 0.0 0.0 0.0
0.0 /base_link /IMU_link 10"/>
</launch>
```

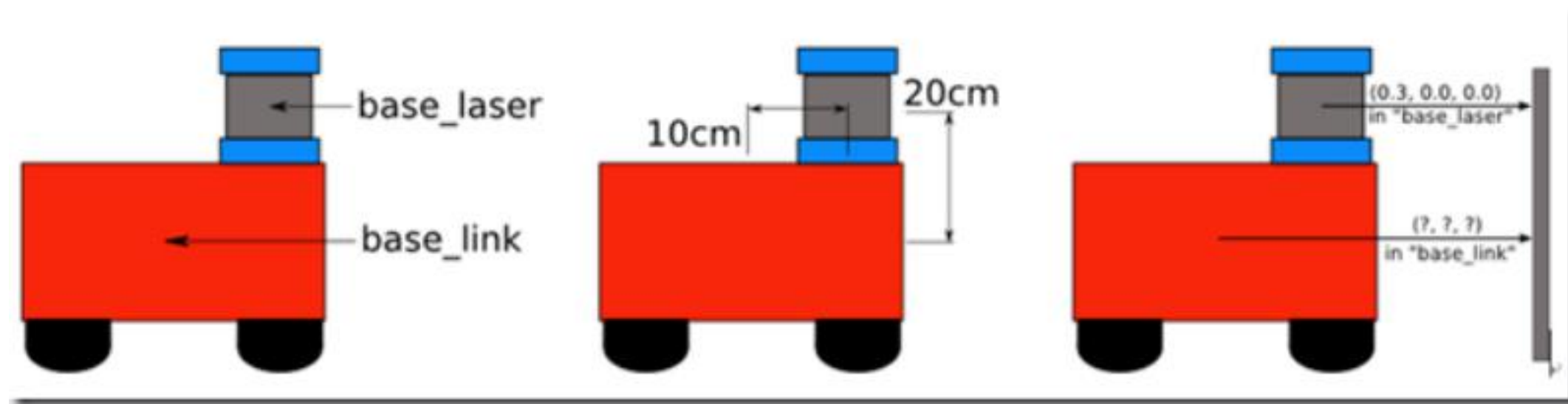
tf 变换

tf 变换具体原理。

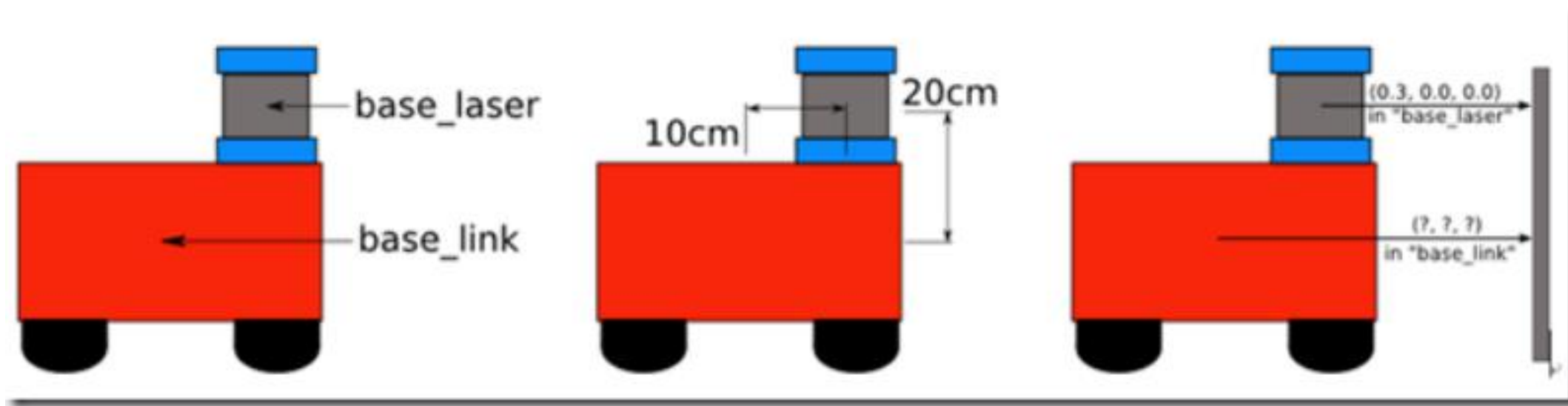


## 设置机器人TF坐标系

~~~~~



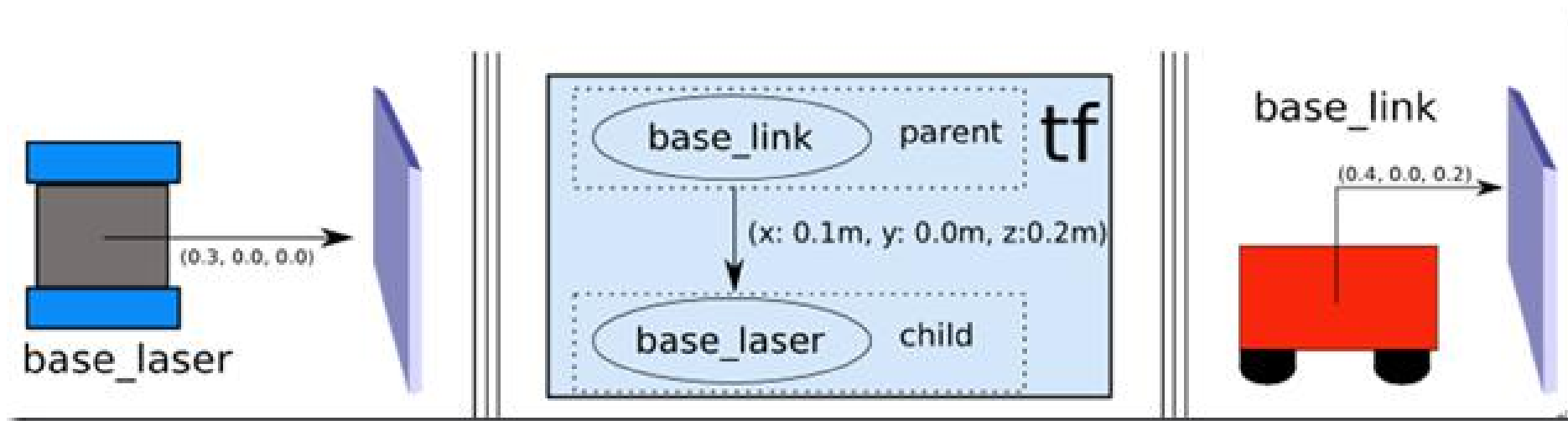
## 设置机器人TF坐标系



$(x: 0.1\text{m}, y: 0.0\text{m}, z: 0.2\text{m}) \rightarrow (x: -0.1\text{m}, y: 0.0\text{m}, z: -0.20\text{m})$



## 设置机器人TF坐标系

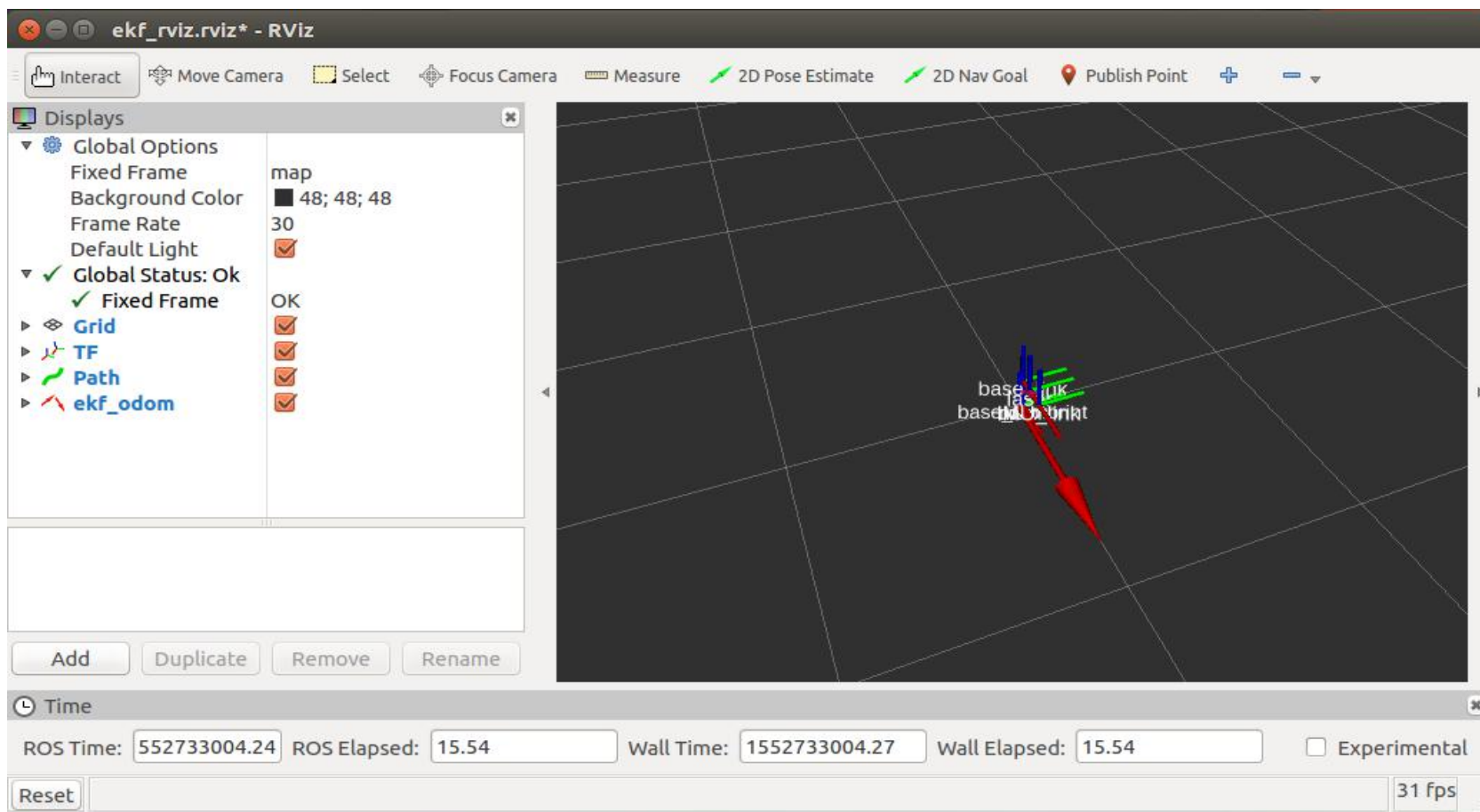


实现激光雷达获取小车轮距和位姿信息之后，会发现小车单独依赖雷达时候会有漂移现象，里程计信息和位姿不太准确，这时候我们就需要加入IMU进行数据融合，是小车实现更加精确的定位。

经过调研选用robot\_localization功能包进行位姿数据的融合，robot\_localization是一系列的机器人状态估计节点集合，其中每一个都是用于三维平面的机器人非线性状态估计，它包括两个机器人状态估计节点ekf\_localization\_node和ukf\_localization\_node。此外也提供了 navsat\_transform\_node节点用于整合GPS数据。

启动ekf\_odom.launch文件进行可视化显示，操作如下：

```
$ roslaunch art_racecar ekf_odom.launch
```







参数整定

```
Run_gmapping.launch (~/.racecar/src/art_racecar/launch) - gedit

Open ▾ [icon] Save

<!-- gmapping -->
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
  <remap from="scan" to="scan"/>
  <param name="map_update_interval" value="2.5"/>
  <param name="maxUrange" value="16.0"/>
  <param name="sigma" value="0.05"/>
  <param name="kernelSize" value="1"/>
  <param name="lstep" value="0.05"/>
  <param name="astep" value="0.05"/>
  <param name="iterations" value="5"/>
  <param name="lsigma" value="0.075"/>
  <param name="ogain" value="3.0"/>
  <param name="lskip" value="0"/>
  <param name="srr" value="0.1"/>
  <param name="srt" value="0.2"/>
  <param name="str" value="0.1"/>
  <param name="stt" value="0.2"/>
  <param name="linearUpdate" value="0.10"/>
  <param name="angularUpdate" value="0.25"/>
  <param name="temporalUpdate" value="1.0"/>
  <param name="resampleThreshold" value="0.25"/>
  <param name="particles" value="30"/>
  <param name="xmin" value="-100.0"/>
  <param name="ymin" value="-100.0"/>
  <param name="xmax" value="100.0"/>
  <param name="ymax" value="100.0"/>
  <param name="delta" value="0.05"/>
  <param name="llsamplerange" value="0.01"/>
  <param name="llsamplestep" value="0.01"/>
  <param name="lasamplerange" value="0.005"/>
  <param name="lasamplestep" value="0.005"/>
  <param name="odom_frame" value="odom"/>
  <param name="base_frame" value="base_footprint"/>
</node>
```

XML ▾ Tab Width: 8 ▾ Ln 16, Col 22 ▾ INS

```
<!-- gmapping -->
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
  <remap from="scan" to="scan" />
  <param name="map_update_interval" value="2.5" />
  <param name="maxUrange" value="16.0" />
```

- map\_update\_interval: 是gmapping过程中每个几秒更新一次地图，该值变小的话表明更新地图的频率加快，会增加消耗更多当前系统的CPU计算资源。同时地图更新也受scanmach的影响，如果scanmatch没有成功的话，不会更新地图。
- maxUrange是雷达可用的最大有效测距值，maxRange是雷达的理论最大测距值，一般情况下设置：  
 $\text{maxUrange} < \text{雷达的现实实际测距值} \leq \text{maxRange}$ 。



- sigma (float, default: 0.05), endpoint匹配标准差
- kernelSize (int, default: 1), 用于查找对应的kernel size
- lstep (float, default: 0.05), 平移优化步长
- astep (float, default: 0.05), 旋转优化步长
- iterations (int, default: 5), 扫描匹配迭代步数
- lsigma (float, default: 0.075), 用于扫描匹配概率的激光标准差
- ogain (float, default: 3.0), 似然估计为平滑重采样影响使用的gain
- lskip (int, default: 0), 每次扫描跳过的光束数.

```
<param name="sigma" value="0.05"/>
<param name="kernelSize" value="1"/>
<param name="lstep" value="0.05"/>
<param name="astep" value="0.05"/>
<param name="iterations" value="5"/>
<param name="lsigma" value="0.075"/>
<param name="ogain" value="3.0"/>
<param name="lskip" value="0"/>
```

## gmapping参数配置

```
<param name="srr" value="0.1"/>
<param name="srt" value="0.2"/>
<param name="str" value="0.1"/>
<param name="stt" value="0.2"/>
<param name="linearUpdate" value="0.10"/>
<param name="angularUpdate" value="0.25"/>
<param name="temporalUpdate" value="1.0"/>
<param name="resampleThreshold" value="0.25"/>
<param name="particles" value="30"/>
```

- srr,srt,str,stt：四个参数是运动模型的噪声参数，一般设置为默认值不用修改。
- linearUpdate：机器人移动多远距离，进行一次 scanmatch 匹配。
- angularUpdate：机器人旋转多少弧度，进行一次scanmatch 匹配。
- temporalUpdate：如果最新扫描处理比更新慢，则处理1次扫描，该值为负数时候关闭基于时间的更新，该参数很重要，需要重点关注。
- resampleThreshold：基于重采样门限的Neff
- particles:gmapping：算法中的粒子数，因为gmapping使用的是粒子滤波算法，粒子在不断地迭代更新，所以选取一个合适的粒子数可以让算法在保证比较准确的同时有较高的速度。

- xmin,ymin,xmax,ymax:初始化的地图大小。
- delta:创建的地图分辨率，默认是0.05m。
- llsamplerange:线速度移动多远距离进行似然估计。
- llsamplestep:线速度移动多少步长进行似然估计。
- lasamplerange:每转动多少弧度用于似然估计。
- lasamplestep:用于似然估计的弧度采样步长是多少。

```
<param name="xmin" value="-100.0"/>  
<param name="ymin" value="-100.0"/>  
<param name="xmax" value="100.0"/>  
<param name="ymax" value="100.0"/>  
<param name="delta" value="0.05"/>
```

```
<param name="llsamplerange" value="0.01"/>  
<param name="llsamplestep" value="0.01"/>  
<param name="lasamplerange" value="0.005"/>  
<param name="lasamplestep" value="0.005"/>
```



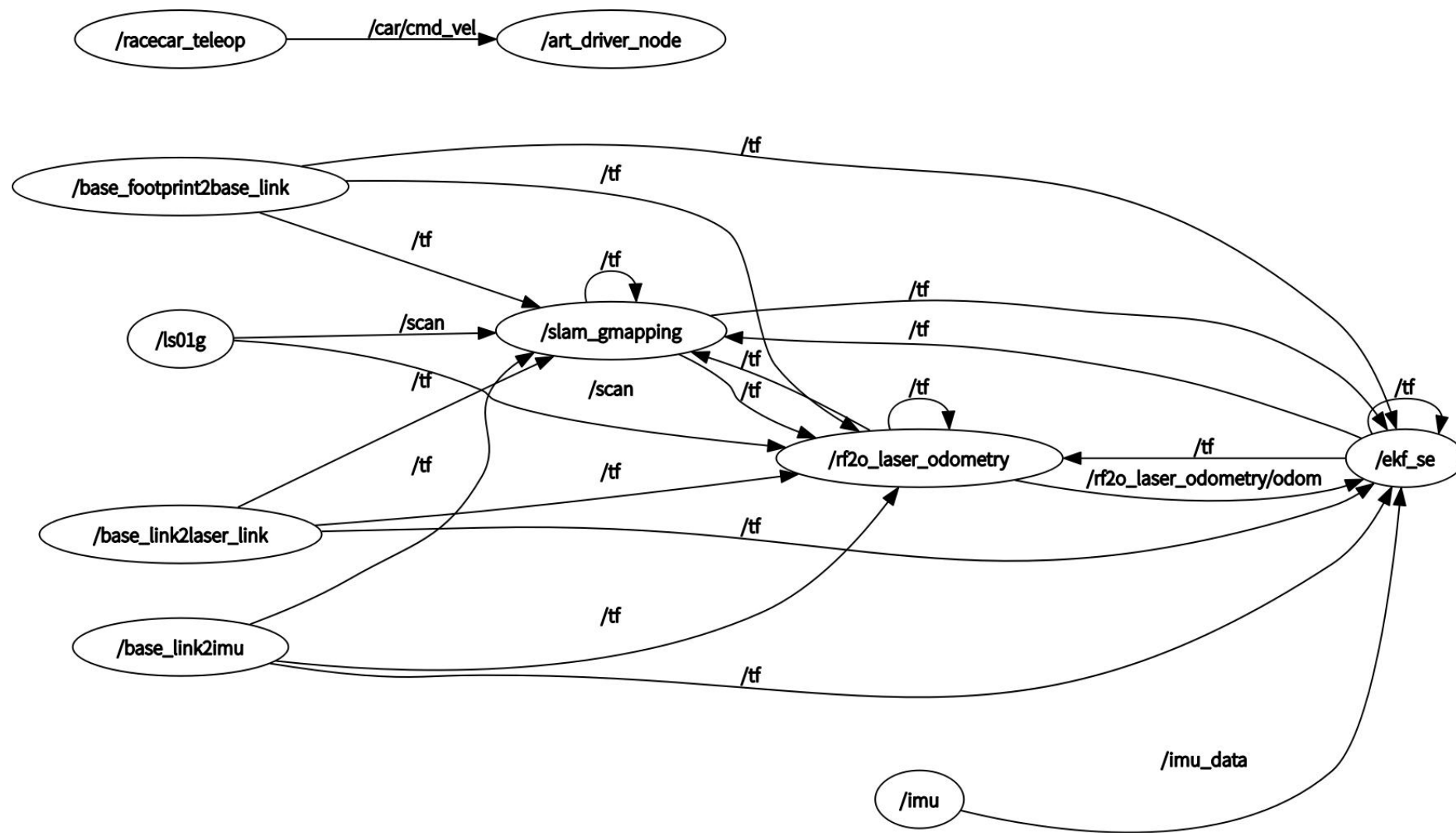
```
<param name="odom_frame" value="odom" />  
<param name="base_frame" value="base_footprint" />
```

- 设置里程计坐标系
- 设置竞速车移动底盘坐标系





# gmapping参数配置





```
<param name="odom_frame" value="odom" />  
<param name="base_frame" value="base_footprint" />
```

- 设置里程计坐标系
- 设置竞速车移动底盘坐标系



~~~~~

```
ubuntu@ubuntu:~$ rostopic list
/accel/filtered
/diagnostics
/imu_data
/map
/map_metadata
/odometry/filtered
/rf2o_laser_odometry/odom
/rosout
/rosout_agg
/scan
/set_pose
/slam_gmapping/entropy
/tf
/tf_static
```

## Amcl-参数配置

global\_costmap\_params.yaml (~/.racecar/src/art\_racecar/param) - gedit

Open ▾



```
global_costmap:
  footprint: [[-0.305, -0.18], [-0.305, 0.18], [0.305, 0.18], [0.305, -0.18]]
  footprint_padding: 0.01
  transform_tolerance: 0.5
  update_frequency: 10.0
  publish_frequency: 10.0

  global_frame: /map
  robot_base_frame: /base_footprint
  resolution: 0.10

  rolling_window: true
  width: 10.0
  height: 10.0
  track_unknown_space: false

  plugins:
    - {name: static, type: "costmap_2d::StaticLayer"}
    - {name: sensor, type: "costmap_2d::ObstacleLayer"}
    - {name: inflation, type: "costmap_2d::InflationLayer"}

  static:
    map_topic: /map
    subscribe_to_updates: true

  sensor:
    observation_sources: laser_scan_sensor
    laser_scan_sensor: {sensor_frame: laser, data_type: LaserScan, topic: scan, marking: true, clearing: true}

  inflation:
    inflation_radius: 0.1
    cost_scaling_factor: 8.0
```



global\_costmap:

footprint: [[-0.305, -0.18], [-0.305, 0.18], [0.305, 0.18], [0.305, -0.18]]

footprint\_padding: 0.01

transform\_tolerance: 0.5

update\_frequency: 10.0

publish\_frequency: 10.0

global\_frame: /map

robot\_base\_frame: /base\_footprint

resolution: 0.10

rolling\_window: true

width: 10.0

height: 10.0

track\_unknown\_space: false



plugins:

- {name: static, type: "costmap\_2d::StaticLayer"}
- {name: sensor, type: "costmap\_2d::ObstacleLayer"}
- {name: inflation, type: "costmap\_2d::InflationLayer"}

static:

map\_topic: /map  
subscribe\_to\_updates: true

sensor:

observation\_sources: laser\_scan\_sensor  
laser\_scan\_sensor: {sensor\_frame: laser, data\_type:  
LaserScan, topic: scan, marking: true, clearing: true}

inflation:

inflation\_radius: 0.1  
cost\_scaling\_factor: 8.0

## Amcl-参数配置

local\_costmap\_params.yaml (~/.racecar/src/art\_racecar/param) - gedit

Open ▾



Save

local\_costmap:

footprint: [[-0.305, -0.18], [-0.305, 0.18], [0.305, 0.18], [0.305, -0.18]]

footprint\_padding: 0.01

transform\_tolerance: 0.5

update\_frequency: 10.0

publish\_frequency: 10.0

global\_frame: /odom

robot\_base\_frame: /base\_footprint

resolution: 0.10

static\_map: true

rolling\_window: true

width: 10.0

height: 10.0

resolution: 0.05

inflation\_radius: 0.1

track\_unknown\_space: false

plugins:

- {name: sensor, type: "costmap\_2d::ObstacleLayer"}
- {name: inflation, type: "costmap\_2d::InflationLayer"}

sensor:

observation\_sources: laser\_scan\_sensor

laser\_scan\_sensor: {sensor\_frame: laser, data\_type: LaserScan, topic: scan, marking: true, clearing: true}

inflation:

inflation\_radius: 0.1

cost\_scaling\_factor: 8.0





local\_costmap:

footprint: [[-0.305, -0.18], [-0.305, 0.18], [0.305, 0.18], [0.305, -0.18]]

footprint\_padding: 0.01

transform\_tolerance: 0.5

update\_frequency: 10.0

publish\_frequency: 10.0

global\_frame: /odom

robot\_base\_frame: /base\_footprint

resolution: 0.10

static\_map: true

rolling\_window: true

width: 10.0

height: 10.0

resolution: 0.05



plugins:

- {name: sensor, type: "costmap\_2d::ObstacleLayer"}
- {name: inflation, type: "costmap\_2d::InflationLayer"}

sensor:

observation\_sources: laser\_scan\_sensor  
laser\_scan\_sensor: {sensor\_frame: laser, data\_type: LaserScan,  
topic: scan, marking: true, clearing: true}

inflation:

inflation\_radius: 0.1  
cost\_scaling\_factor: 8.0



## Amcl-参数配置

~~~~~

```
dwa_local_planner_params.yaml (~/.racecar/src/art_racecar/param) - gedit

Open ▾ [icon]

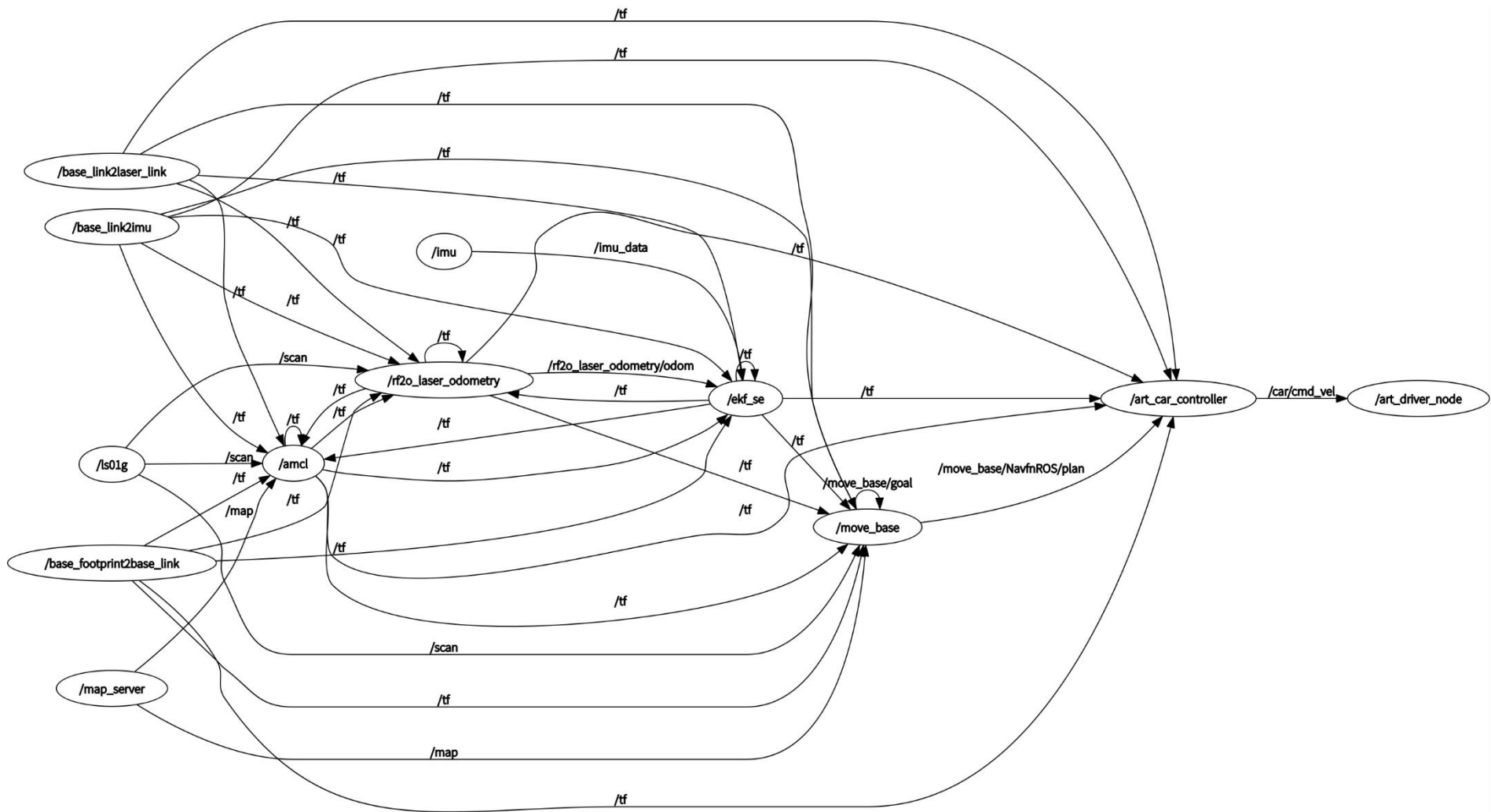
DWAPlannerROS:

  acc_lim_th: 0.1
  acc_lim_x: 0.5
  acc_lim_y: 0.0

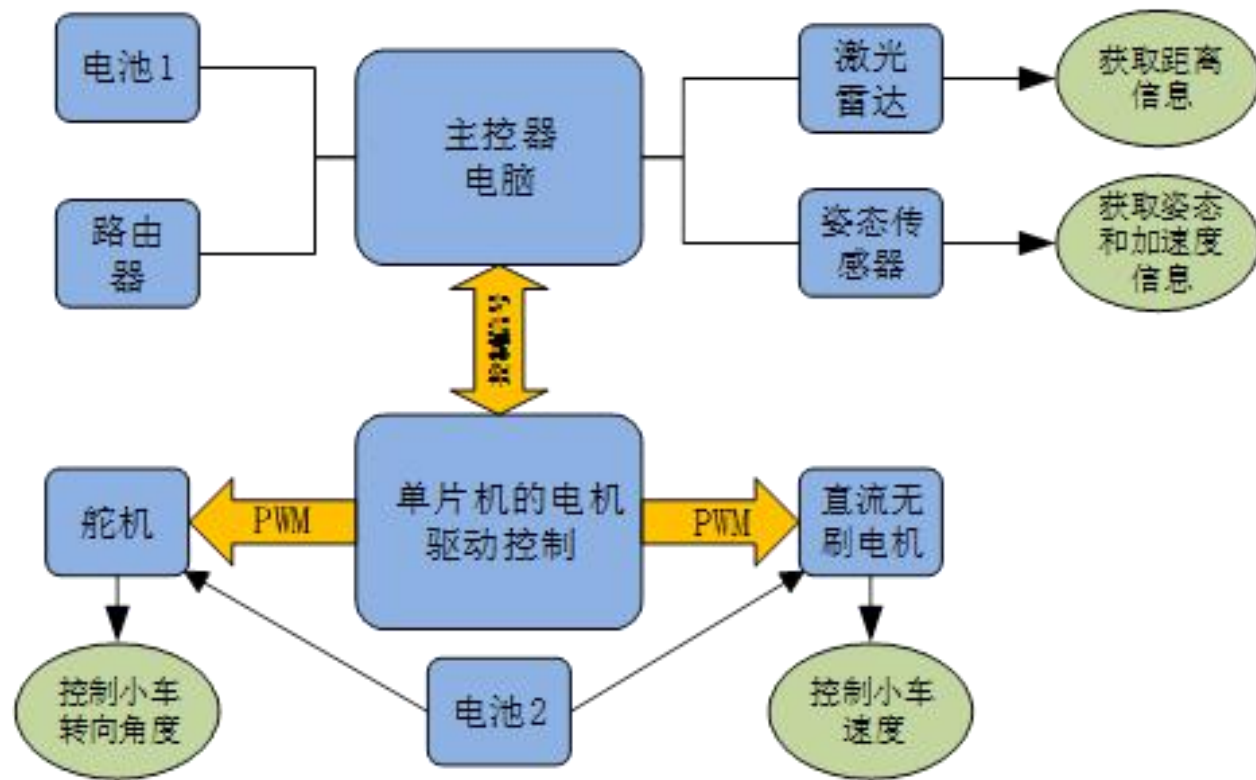
  max_vel_x: 0.5
  min_vel_x: 0.0

  max_vel_y: 0.0
  min_vel_y: 0.0

  max_trans_vel: 0.5
  min_trans_vel: 0.1
  max_rot_vel: 0.2
  min_rot_vel: 0.0
```



□ 软件系统部分如下图所示，机器人操作系统ROS通过外部激光雷达和IMU姿态传感器获取无人竞速车周围环境信息与车体的姿态信息，将这些采集到的数据进行处理，通过SLAM算法实现对整个环境的地图构建，有了地图之后，我们就可以通过AMCL等算法实现地图上的路径规划和自主导航。





谢谢！