DynamicX培训
ROS导论 —— 第二课

广东工业大学DynamicX 机器人队

第二课概要

- ROS 软件包结构
- 使用 CLion 开发 ROS 程序
- ROS C++ 客户端库 (roscpp)
- ROS subscribers 和 publishers
- ROS 参数服务器
- RViz 可视化界面

ROS pakages

- ROS 软件被组织成包，其中可以包含源代码，launch文件，配置文件、消息定义、数据和文档

- 编译包如果需要其他软件包(例如:消息定义)，声明这些为依赖项

创建软件包:

```
>  catkin create pkg {package_name}
--catkin-deps {dependencies}
```

ROS Packages

package.xml

- package.xml 定义了包的基本信息
- 包名
- 版本号
- 作者
- 依赖的包
- …

More info
wiki.ros.org/catkin/package.xml

Packages.xml

```
<xml version="1.0">
<package format="2">
   <name>ros_package_template</name>
   <version>0.1.0</version>
   <description>A ROS package that. . .</description>
   <maintainer email="tlankhorst6ethz.ch">Tom Lankhorst</maintainer>
   <license>BSD</license>
   <url
   type="website">https://github.com/leggedrobotics/ros_.</url>

   <author email="tlankhorst6ethz.ch">Tom Lankhorst</author>

   <buildtool_depend>catkin</buildtool_depend>

   <depend>roscpp</depend>
   <depend>std_msgs</depend>

   <build_depend>message_generation</build_depend>
</package>
```

ROS Packages

CMakeLists.txt

- 要求的 CMake 版本 (cmake_minimum_required)
- 包名(project())
- 配置 C++ 编译标准特性
- 查找编译所需的其他 CMake/Catkin 包 (find_package()).
- Message/Service/Action Generators(add_message_files(), add_service_files(), add_action_files())
- 涉及的 message/service/action generation (generate_messages())
- 为构建系统指定特定的 catkin 信息 (catkin_package())

CMakeList.txt

```
cmake_minimum_required(VERSION 3.10.2)
project(ros_package_template)

## Use C++14, or 11...
set(CMAKE_CXX_STANDARD 14)
set(CMAKE_CXX_STANDARD_REQUIRED TRUE )

## Find catkin macros and libraries

find_package(catkin REQUIRED
          COMPONENTS
          roscpp
          sensor_msgs)
```

More info
wiki.ros.org/catkin/CMakeLists.txt

ROS Packages

CMakeLists.txt 实例

```
cmake_minimum_required(VERSION 3.10.2)
project(smb_highlevel_controller)

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED TRUE)

find_package(catkin REQUIRED
      COMPONENTS roscpp
      sensor_msgs
)
catkin_package(
       INCLUDE_DIRS include
       # LIBRARIES
       CATKIN_DEPENDS roscpp sensor_msgs
       # DEPENDS
)

include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(${PROJECT_NAME}
src/${PROJECT_NAME}_node.cpp
src/SmbHighlevelController.cpp)

target_link_libraries(${PROJECT_NAME} ${catkin_LIBRARIES})
```

在 package.xml 中编写相同的包名

默认使用 C++11 (or 14)

列出编译软件包所需的包（必须在 package.xml 中列出）

指定编译信息

- INCLUDE_DIRS: 软件包导出的头文件路径
- LIBRARIES: 在此项目中导出的库
- CATKIN_DEPENDS: 指定依赖于此包的 catkin 依赖项。
- DEPENDS: 依赖于该包的非 catkin 依赖项(需要在 package.xml 列举)

指明头文件的路径

声明为一个可执行文件，这个节点带有两个 src 文件

指定链接可执行文件的库

使用 CLion 开发 ROS 程序

- 在工作空间中创建软件包

```
> cd rm_ws
> cd src
> catkin create pkg my_package --roscpp rospy
std_msgs
```

- 回到工作空间并配置环境

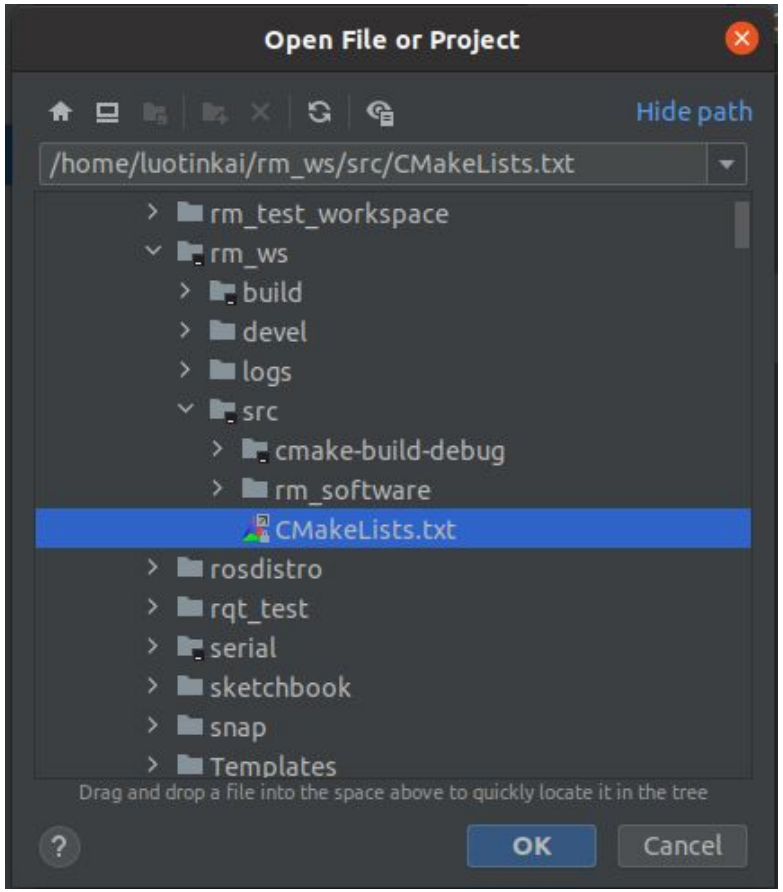
```
> cd ..
> source ./devel/setup.bash
```

使用 CLion 开发 ROS 程序

- 在终端启动 Clion

```
> sh YOUR_CLION/bin/clion.sh
```

- 点击 CMakeList.txt 打开项目



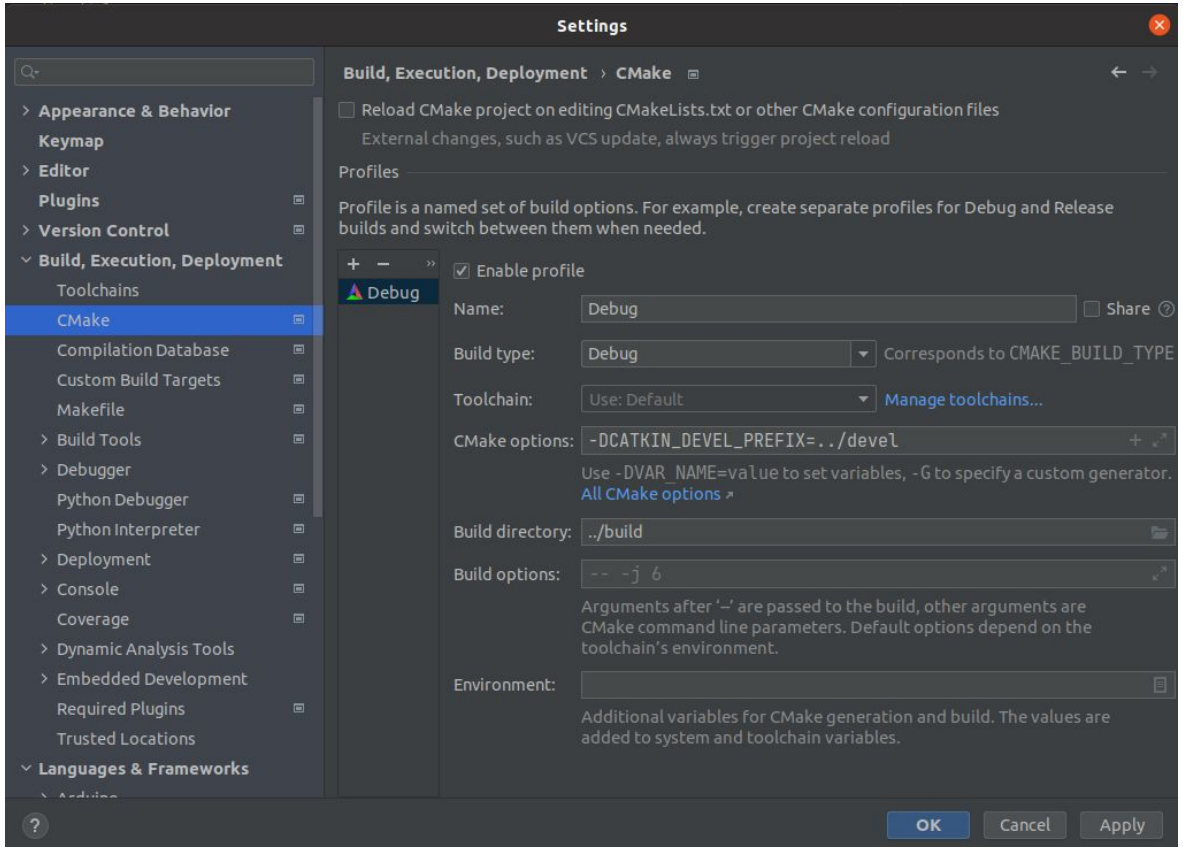
使用 CLion 开发 ROS 程序

- 配置 CMake
  - CMake options

```
-DCATKIN_DEVEL_PREFIX=../devel
```

  - Build options

```
../build
```



ROS C++ Client Library (roscpp）

客户端库的基本构成

- 初始化
- NodeHandle
- logging
- pubulisher / suscriber
- 参数服务器

在第四课中所涉及到的东西

- 服务
- 动作
- 时间

ROS C++ Client Library (roscpp)

初始化

hello_world.cpp

```
#include <ros/ros.h>


int main(int argc, char* argv[])
{
  ros::init(argc, argv, "hello_world");
  ros::NodeHandle nodeHandle;
  ros::Rate loopRate(10);

  unsigned int count = 0;
  while (ros::ok()) {

  ROS_INFO_STREAM("Hello World " << count);

  ros::spinOnce();
  loopRate.sleep();
  count++;
}
  return 0;
}
```

ROS 主要包含的头文件

ros::init(…) 会在其他 ros 函数之前被调用

节点句柄是与 ROS 系统通信的接入点（主题、服务、参数）

ros::Rate 用于以所需的频率运行循环

ros::ok() 检查节点是否应继续运行，当接收到 SIGINT (Ctrl + C) 或者 ros::shutdown() 被调用的时候返回 false

ROS_INFO() ROS日志文件信息

ros::spinOnce() 处理传入消息的回调函数

More info

wiki.ros.org/roscpp

wiki.ros.org/roscpp/Overview

ROS C++ Client Library (roscpp)

Node Handle

这里有四种节点句柄命名空间的种类

- node handle 默认的命名空间:
  nh_ = ros::NodeHandle();
- node handle 私有命名空间:
  nh_private_ = ros::NodeHandle("~");
- node handle 指定命名空间:
  nh_gdut_ = ros::NodeHandle("gdut");
- node handle 全局命名空间:
  nh_global_ = ros::NodeHandle("/");

推荐

不推荐

- 这些节点的命名空间将解析为:
  /namespace/topic
  /namespace/node/topic
  /namespace/gdut/topic
  /topic

More info
wiki.ros.org/roscpp/Overview/NodeHandles

ROS C++ Client Library (roscpp)

Subscriber

- 当收到消息时，回调函数就会被调用并且收到的消息会作为参数
- 通过调用节点句柄的 subscribe() 方法来订阅话题

```
ros::Subscriber subscriber =
nodeHandle.subscribe(topic,queue_size,
callback_function);
```

- 持续订阅话题，直到你取消订阅
- ros:: spin() 处理回调函数，直到节点关闭后才会返回

listener.cpp

```
#include <ros/ros.h>
#include <std_msgs/String.h>

void chatterCallback(const std_msgs::String& msg)
{
ROS_INFO("I heard: [%s]", msg.data.c_str());
}

int main(int argc, char* argv[])
{
   ros::init(argc ,argv, "listener");
   ros::NodeHandle nodeHandle;

   ros:Subscriber subscriber =
   nodeHandle.subscribe("chatter",10, chatterCallback);
   ros::spin();
   return 0;
}
```

More info

wiki.ros.org/roscpp/Overview/Parameter%20Server

ROS C++ Client Library (roscpp)

Logging

- 用于节点中记录可读的文本的控制台和日志文件
- 使用例如ROS_INFO代替std::cout
- 自动记录到控制台、日志文件和/rosout 话题
- 不同的严重性级别(信息、警告等)
- 支持printf- stream-style 格式

```
ROS_INFO("Result: %d", result); //printf
ROS_INFO_STREAM("Result: " << result);
```

- 其他功能，如状况、限制、延迟记录等.

	Debug	Info	Warn	Error	Fatal
stdout	✓	✓			
stderr			✓	✓	✓
Log file	✓	✓	✓	✓	✓
/rosout	✓	✓	✓	✓	✓

! 要在控制台中查看输出，请在 launch 文件中将输出配置设置为screen

```
<launch>
    <node name="listener" ... output="screen"/>
</launch>
```

More info
wiki.ros.org/roscpp/Overview/NodeHandles

ROS C++ Client Library (roscpp)

publisher

- 用句柄创建 publisher

```
ros::Publisher publisher =
nodeHandle.advertise<message_type>(topic
, queue_size);
```

- 编写消息内容
- 发布消息

```
publisher.publish(message);
```

More info

wiki.ros.ora/roscpp/Overview/Publishers%2Qand%2QSubscribers

talker.cpp

```
#include <ros/ros.h>
#include <std_msgs/String.h>

int main(int argc, char* argv[]) {

      ros::init(argc, argv, "talker");
      ros::NodeHandle nh;
      ros::Publisher chatterPublisher =
      nh.advertise<std_msgs::String>("chatter", 1);
      ros::Rate loopRate(10);

      unsigned int count = 0;

      while (ros::ok()) {
      std_msg::String message;
      message.data = "hello world " + std::to_string(count);
      ROS_INFO_STREAM(message.data);
      chatterPublisher.publish(message);
      ros::spinOnce();
      loopRate.sleep();
      count++;
}
      return 0;
}
```

ROS 参数服务器

- 节点使用参数服务器来存储参数以及在运行时检索参数
- 最好用于静态数据，例如配置参数
- 参数可以在启动文件或单独的YAML文件中定义

列举参数服务器中的参数

```
> rosparam list
```

给参数服务器中的参数设置大小

```
> rosparam get parameter_name
```

从参数服务器中取到参数的值

```
> rosparam set parameter_name value
```

config.yaml

```
camera:
  left:
    name: left_camera
    exposure: 1
  right:
    name: right_camera
    exposure: 1.1
```

package.launch

```
<launch>
    <node name="name" pkg="package" type="node_type">
    <rosparam command="load"
              file="$(find package)/config/config.yaml" />
  </node>
</launch>
```

More info

wiki.ros.org/rosparam

ROS C++ Client Library (roscpp)

参数

- 使用 C++ 来获取参数

```
nodeHandle.getParam(parameter_name, variable)
```

```
ros::NodeHandle nodeHandle("~");
std::string topic;
if (nodeHandle.getParam("topic", topic)) {
ROS_ERROR("Could not find topic parameter!");
}
ROS_INFO_STREAM("Read topic: " << topic);
```

- 当函数找到对应的参数就会返回 true 否则返回false。
- 全局和局部参数访问:
  - 全局参数

```
nodeHandle.getParam(“/package/camera/left/exposure”,variable)
```

  - 局部参数 (与节点句柄相关)

```
nodeHandle.getParam(“camera/left/exposure”, variable)
```

- 对于参数，通常使用专用节点句柄 ros::NodeHandle(“~”)

More info

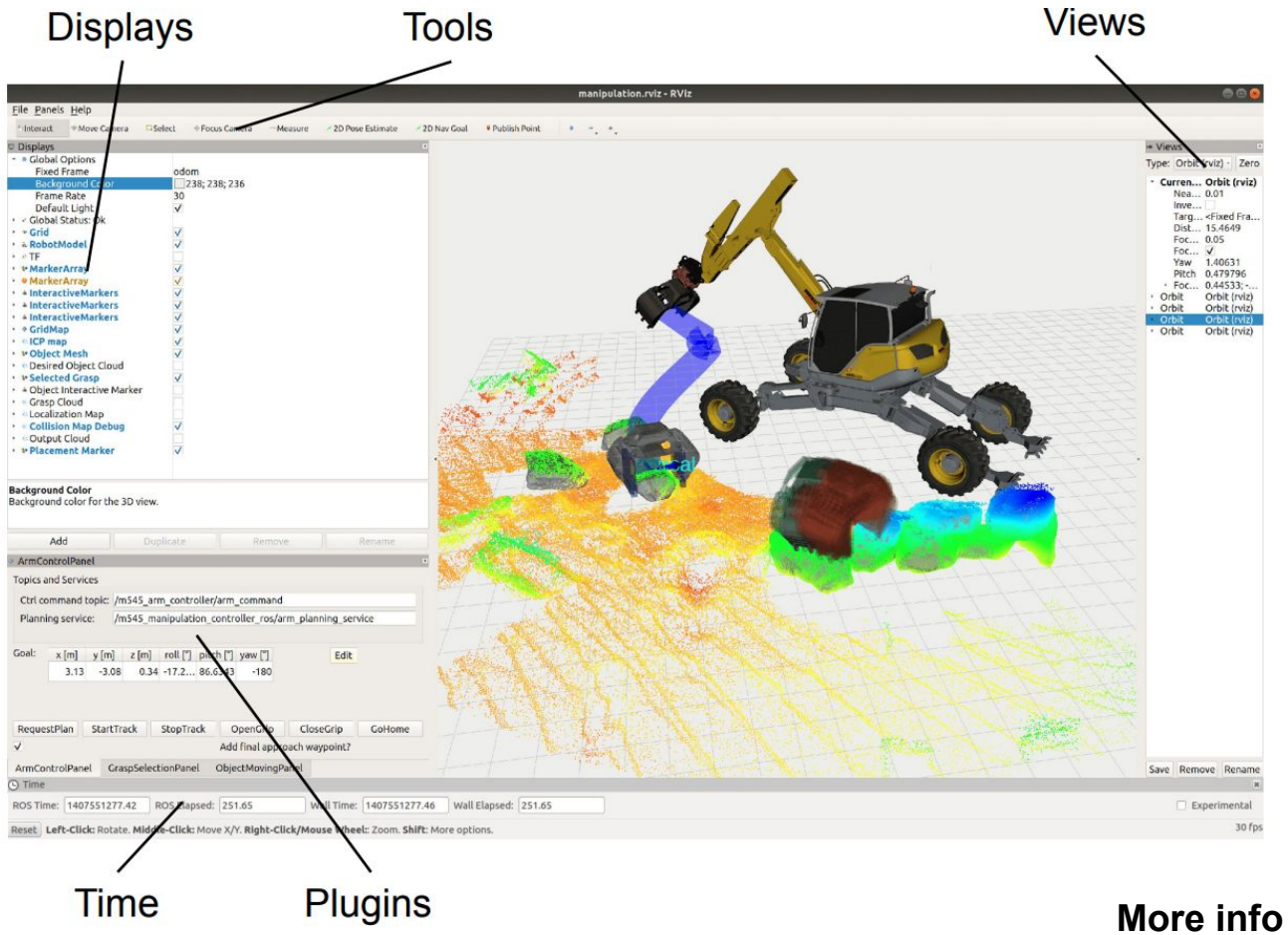
wiki.ros.org/roscpp/Overview/Parameter%20Server

RViz

- ROS 的 3D 可视化工具
- 订阅节点并查看消息内容
- 不同的摄像机图像 (orthographic, top-down, etc.)
- 发送用户信息的交互工具
- 将设置保存并加载为RViz配置
- 可扩展插件

运行 Rviz:

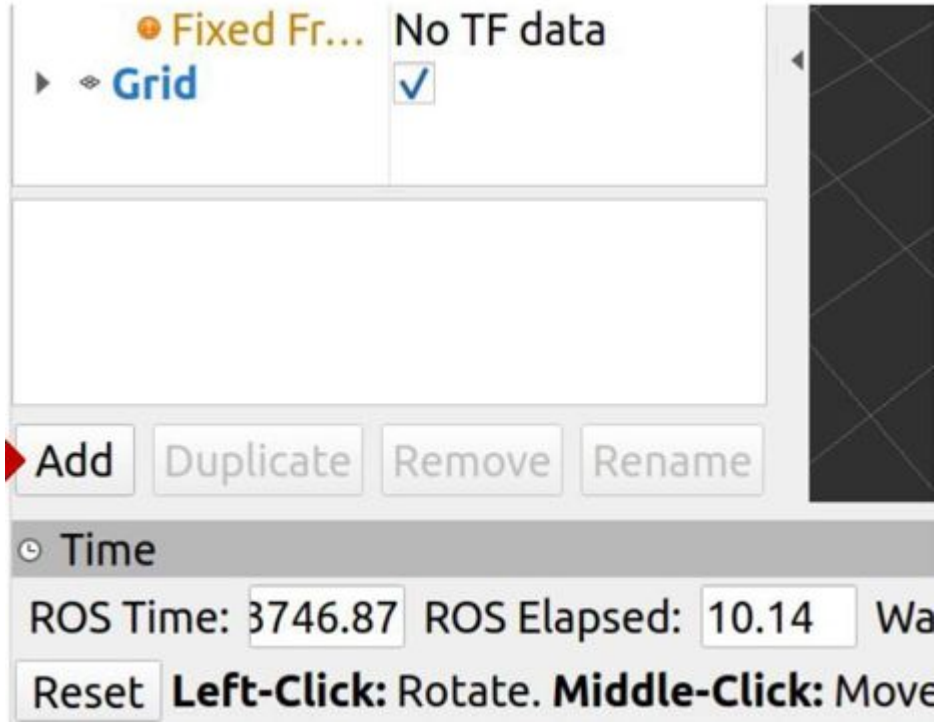
```
> rviz
```

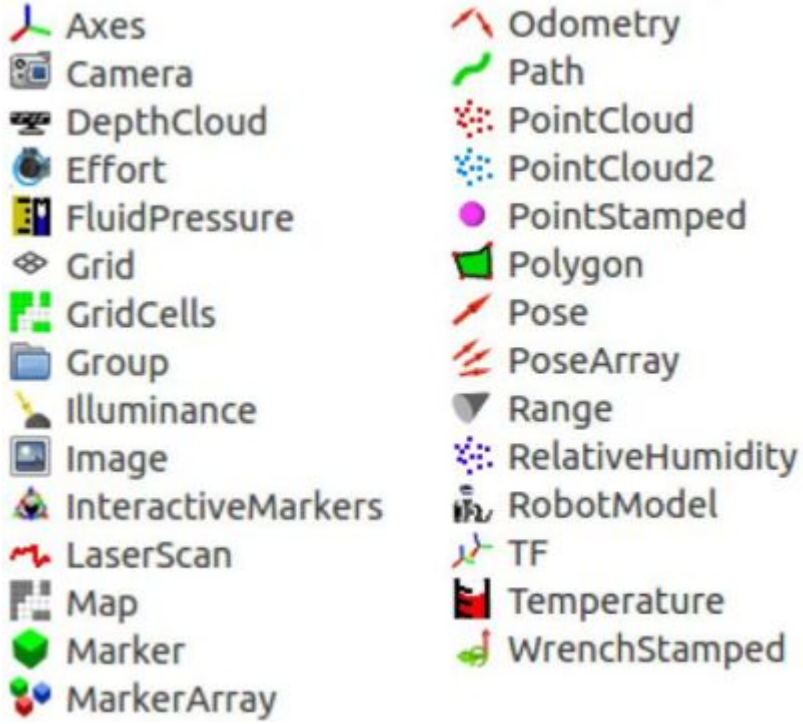


More info
wiki.ros.org/rviz

RViz

可视化插件





使用 Ctrl + S 来保存配置

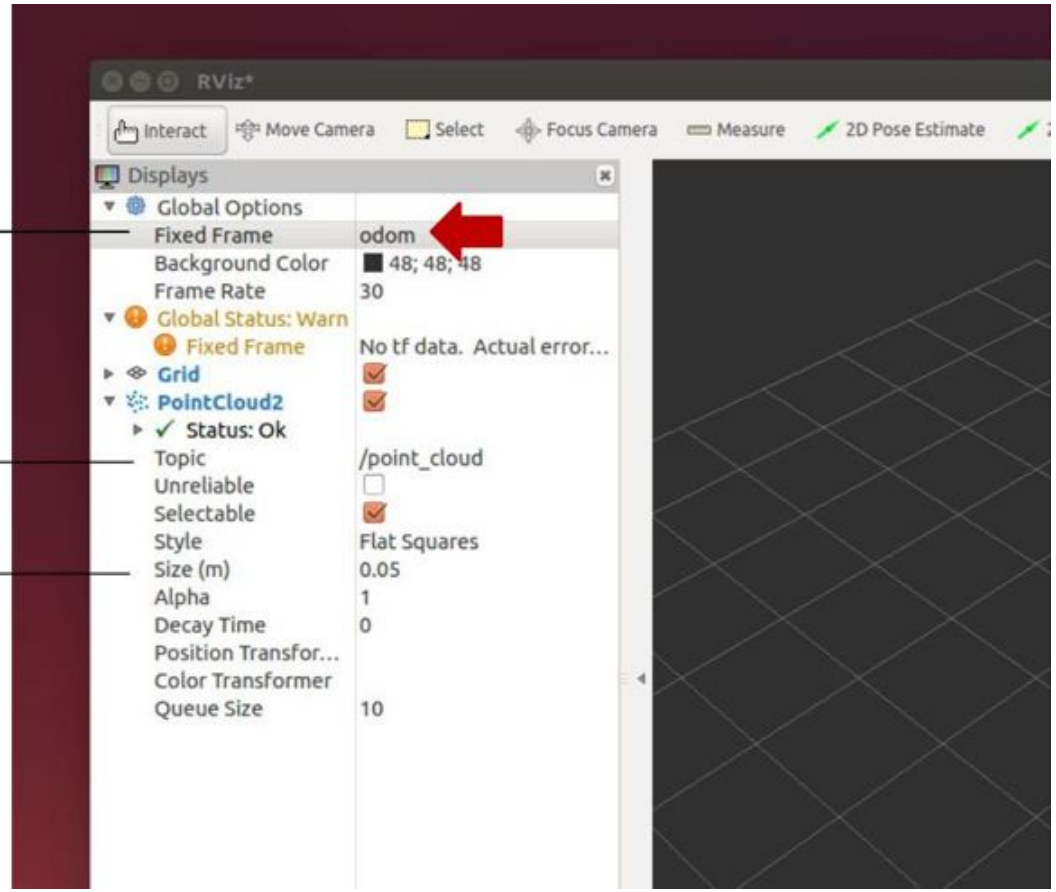
RViz

实例

! 设置固定坐标系(数据必须是存在的！)

选择显示的话题

更改显示选项(例如大小)



更多资料

- ROS Wiki
  - https://wiki.ros.org/
- Installation
  - https://wiki.ros.org/ROS/Installation
- Tutorials
  - https://wiki.ros.org/ROS/Tutorials
- Available packages
  - https://www.ros.org/browse/

- ROS Best Practices
  - https://github.com/leggedrobotics/ros_best_practices/wiki
- ROS Package Template
  - https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template
- ROS Cheat Sheet
  - https://kapeli.com/cheat_sheets/ROS.docset/ Contents/Resources/Documents/index