深蓝学院 shenlanxueyuan.com

古月居 GYH.AI

ROS理论与实践

—— 第4讲：ROS常用组件工具

主讲人 胡春旭

机器人博客“古月居”博主

《ROS机器人开发实践》作者

武汉精锋微控科技有限公司 联合创始人

华中科技大学 人工智能与自动化学院 硕士

目录

古月居 GYH.AI

1. Launch启动文件

1. Launch启动文件

古月居 GYH.AI

```
<launch>
  <!-- local machine already has a definition by default.
       This tag overrides the default definition with
       specific ROS_ROOT and ROS_PACKAGE_PATH values -->
  <machine name="local_alt" address="localhost" default="true" ros-root="/u/user/ros/ros/" ros-package-path="/u/user/ros/ros-pkg" />
  <!-- a basic listener node -->
  <node name="listener-1" pkg="rospy_tutorials" type="listener" />
  <!-- pass args to the listener node -->
  <node name="listener-2" pkg="rospy_tutorials" type="listener" args="-foo arg2" />
  <!-- a respawn-able listener node -->
  <node name="listener-3" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start listener node in the 'wg1' namespace -->
  <node ns="wg1" name="listener-wg1" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start a group of nodes in the 'wg2' namespace -->
  <group ns="wg2">
    <!-- remap applies to all future statements in this scope. -->
    <remap from="chatter" to="hello"/>
    <node pkg="rospy_tutorials" type="listener" name="listener" args="--test" respawn="true" />
    <node pkg="rospy_tutorials" type="talker" name="talker">
      <!-- set a private parameter for the node -->
      <param name="talker_1_param" value="a value" />
      <!-- nodes can have their own remap args -->
      <remap from="chatter" to="hello-1"/>
      <!-- you can set environment variables for a node -->
      <env name="ENV_EXAMPLE" value="some value" />
    </node>
  </group>
</launch>
```

Launch文件：通过XML文件实现多节点的配置和启动（**可自动启动ROS Master**）

1. Launch启动文件

```
<launch>
    <node pkg="learning_communication" type="person_subscriber" name="talker" />
    <node pkg="learning_communication" type="person_publisher" name="listener" />
</launch>
```

<launch> launch文件中的根元素采用<launch>标签定义

<node>

启动节点

<node pkg="package-name" type="executable-name" name="node-name" />

- pkg：节点所在的功能包名称
- type：节点的可执行文件名称
- name：节点运行时的名称
- output、respawn、required、ns、args

1. Launch启动文件

古月居 GYH.AI

参数设置

<param> / <rosparam>

设置ROS系统运行中的参数，存储在参数服务器中。

```
<param name="output_frame" value="odom"/>
```

- name：参数名
- value：参数值

加载参数文件中的多个参数：

```
<rosparam file="params.yaml" command="load" ns=“params" />
```

<arg>

launch文件内部的局部变量，仅限于launch文件使用

```
<arg name="arg-name"  default="arg-value" />
```

- name：参数名
- value：参数值

调用：

```
<param name="foo" value="$(arg arg-name)" />
<node name="node" pkg="package" type="type " args="$(arg arg-name)" />
```

1. Launch启动文件

古月居 GYH.AI

重映射

<remap >

重映射ROS计算图资源的命名。

```
<remap from="/turtlebot/cmd_vel" to="/cmd_vel"/>
```

- from：原命名
- to：映射之后的命名

嵌套

<include>

包含其他launch文件，类似C语言中的头文件包含。

```
<include file="$(dirname)/other.launch" />
```

- file：包含的其他launch文件路径

* 更多标签可参见：http://wiki.ros.org/roslaunch/XML

1. Launch启动文件

```
<launch>

    <param name="/turtle_number"    value="2"/>
    <arg name="TurtleName1"  default="Tom" />
    <arg name="TurtleName2"  default="Jerry" />

    <node pkg="turtlesim" type="turtlesim_node" name="turtlesim_node">
        <param name="turtle_name1"    value="$(arg TurtleName1)"/>
        <param name="turtle_name2"    value="$(arg TurtleName2)"/>

        <rosparam file="$(find learning_launch)/config/param.yaml" command="load"/>
    </node>

    <node pkg="turtlesim" type="turtle_teleop_key" name="turtle_teleop_key" output="screen"/>

</launch>
```

turtlesim_parameter_config.launch

1. Launch启动文件

```
<launch>

    <include file="$(find learning_launch)/launch/simple.launch" />

    <node pkg="turtlesim" type="turtlesim_node" name="turtlesim_node">
        <remap from="/turtle1/cmd_vel" to="/cmd_vel"/>
    </node>

</launch>
```

turtlesim_remap.launch

古月居
GYH.AI

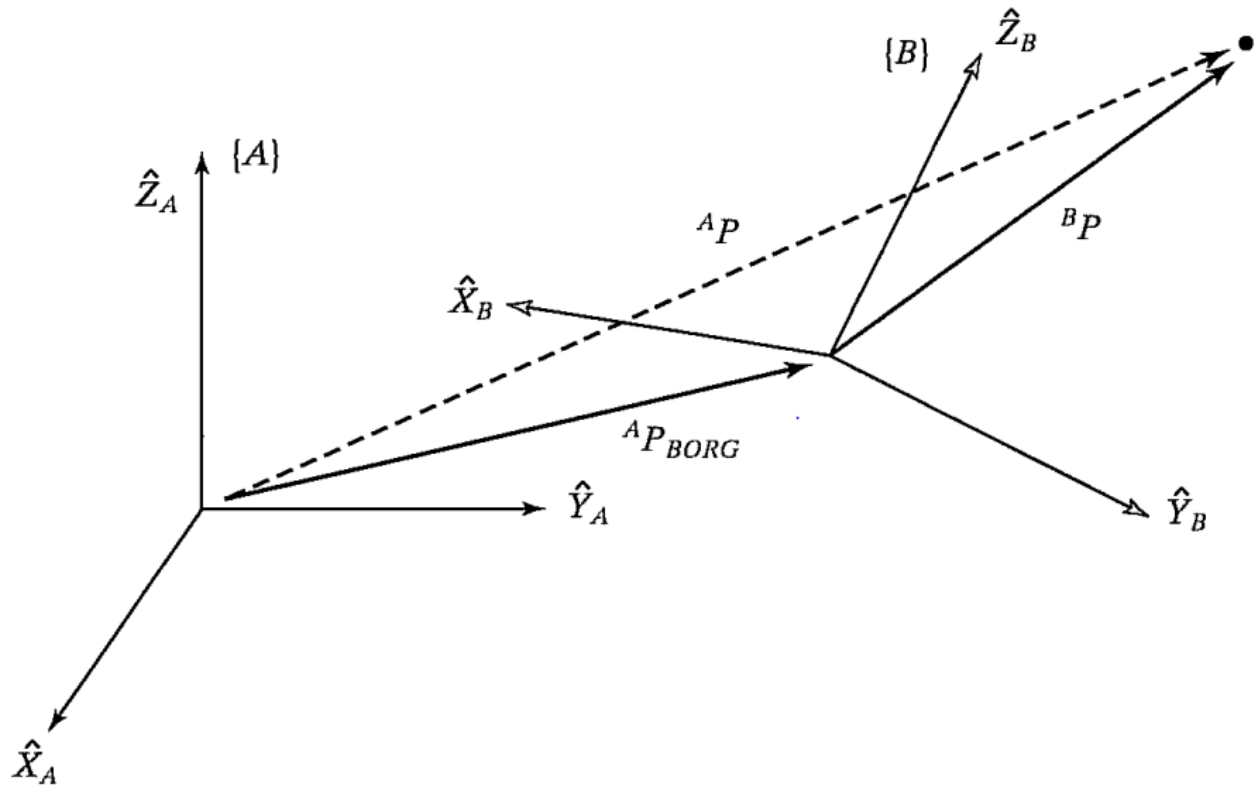
2. TF坐标变换

2. TF坐标变换

$$ {}^{A}P = {}^{A}_{B}R\,{}^{B}P + {}^{A}P_{BORG}. $$

$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}P. $$

$$ \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}. $$



某位姿在A、B两个坐标系下的坐标变换

* 参考：《机器人学导论》

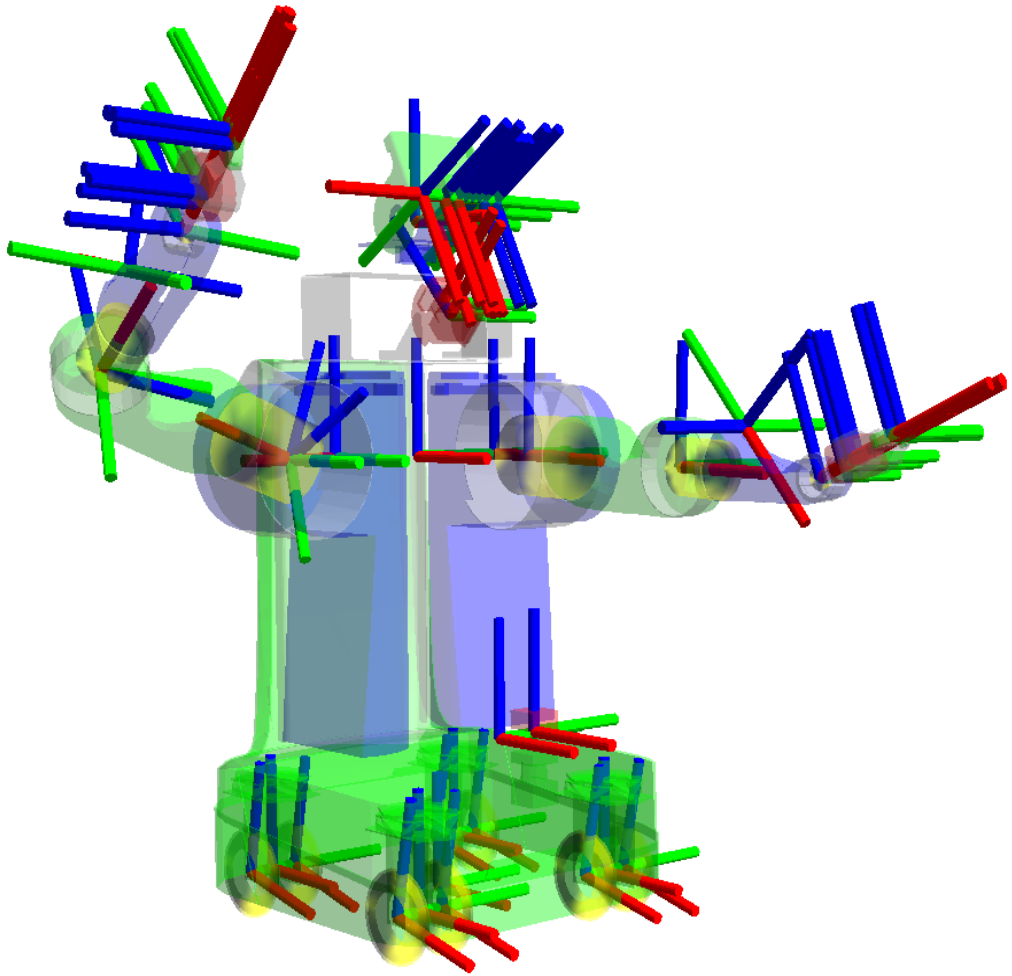
2. TF坐标变换

TF功能包能干什么?

- 五秒钟之前，机器人头部坐标系相对于全局坐标系的关系是什么样的?
- 机器人夹取的物体相对于机器人中心坐标系的位置在哪里?
- 机器人中心坐标系相对于全局坐标系的位置在哪里?

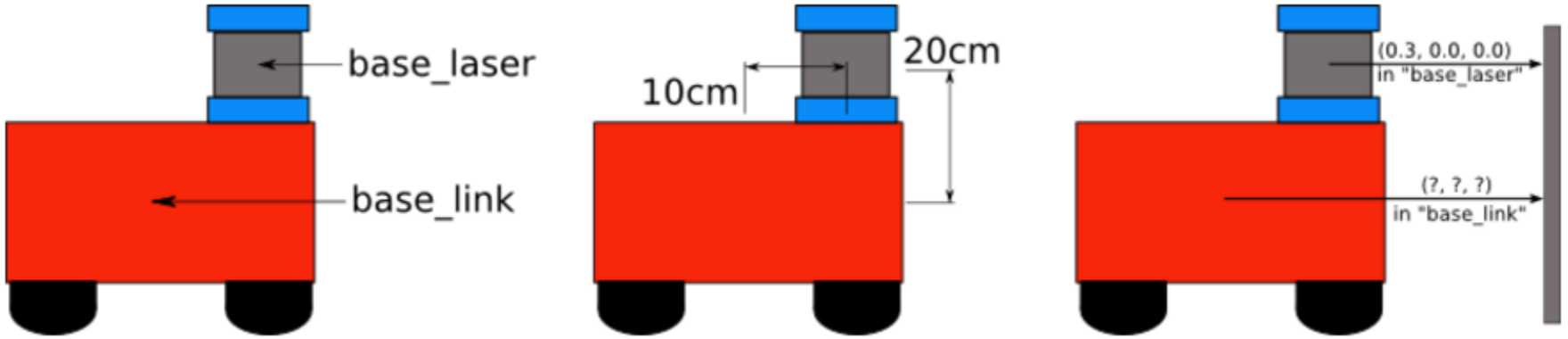
TF坐标变换如何实现?

- 广播TF变换
- 监听TF变换

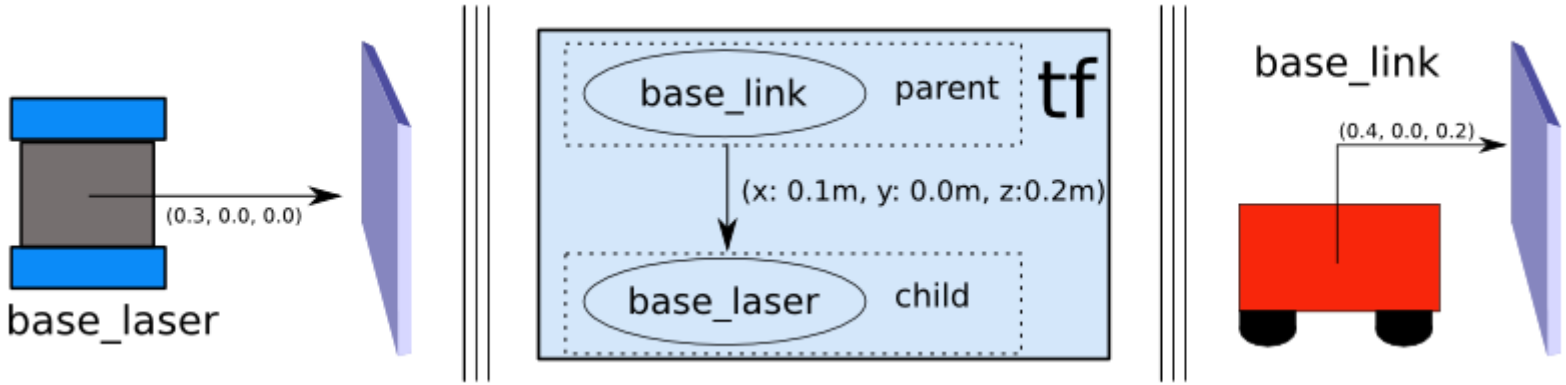


机器人系统中繁杂的坐标系

2. TF坐标变换



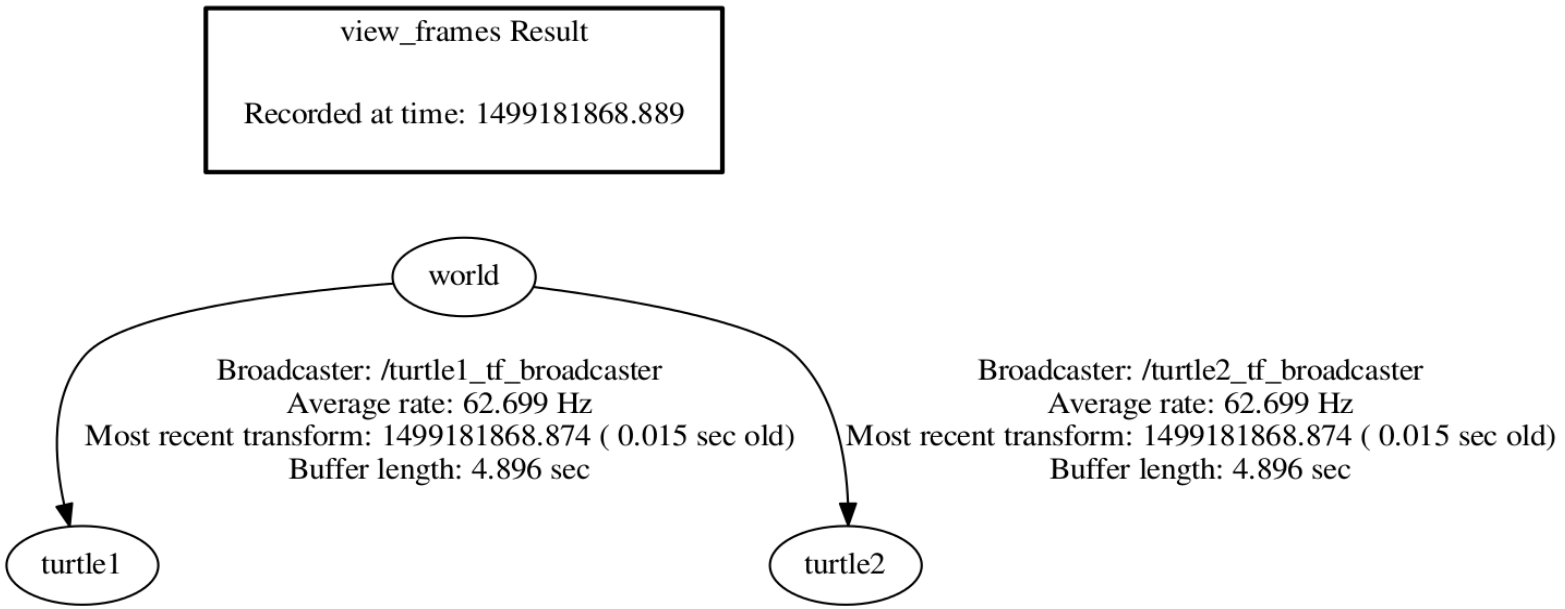
移动机器人的本体坐标系与雷达坐标系

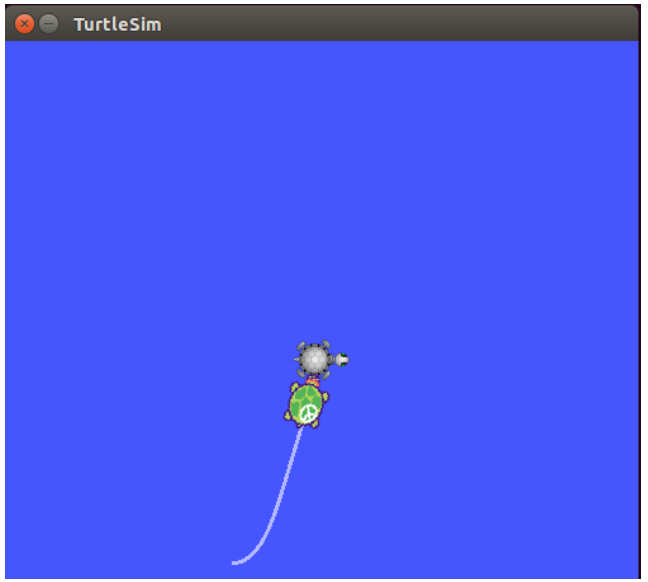


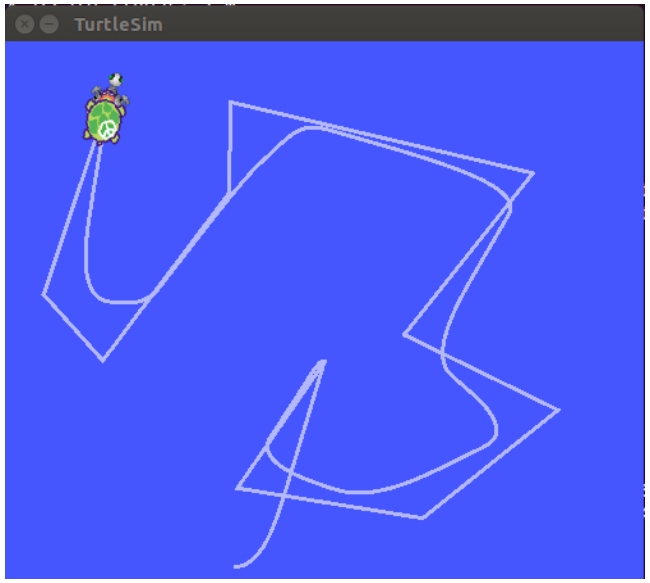
坐标系之间的数据变换

2. TF坐标变换

```
$ sudo apt-get install ros-melodic-turtle-tf
$ roslaunch turtle_tf turtle_tf_demo.launch
$ rosrun turtlesim turtle_teleop_key
$ rosrun tf view_frames
```







小海龟跟随实验

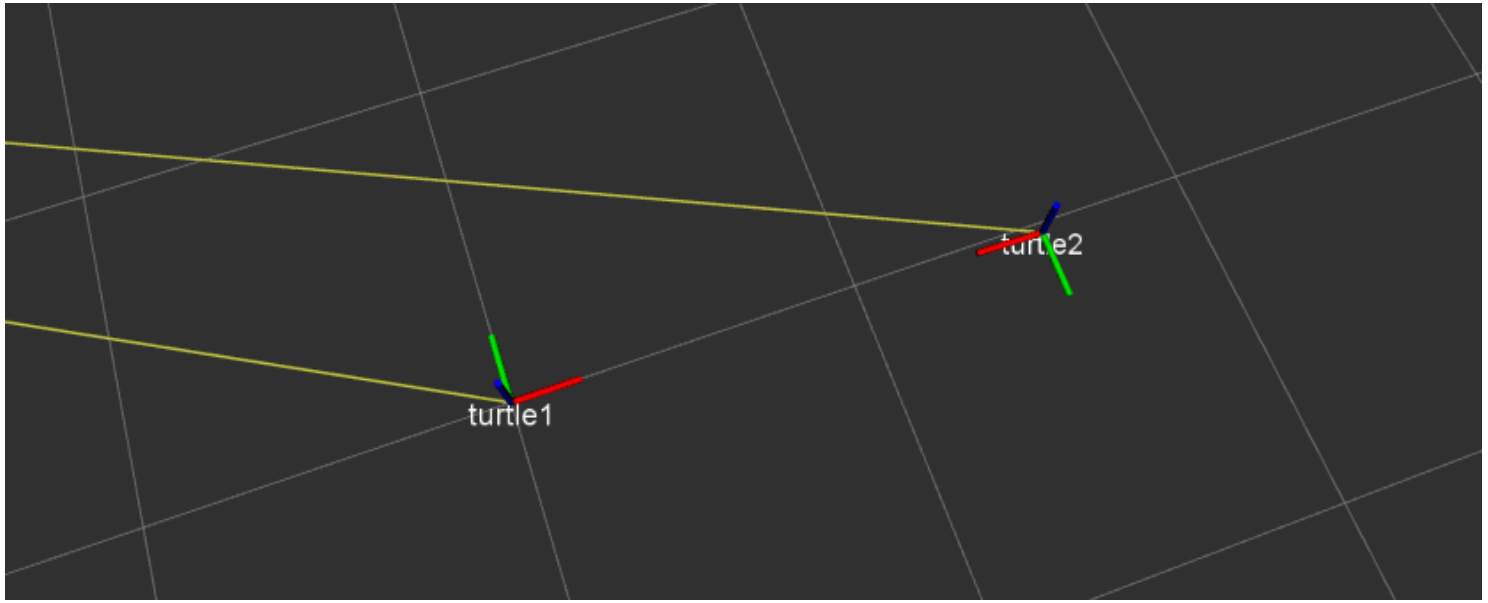
2. TF坐标变换

命令行工具

```
➜ ~ rosrun tf tf_echo turtle1 turtle2
At time 1504942486.329
- Translation: [0.000, 0.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.311, 0.950]
            in RPY (radian) [0.000, -0.000, 0.633]
            in RPY (degree) [0.000, -0.000, 36.290]
At time 1504942487.018
- Translation: [0.000, 0.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.311, 0.950]
            in RPY (radian) [0.000, -0.000, 0.633]
            in RPY (degree) [0.000, -0.000, 36.290]
```

$$T_{turtle1\_turtle2} = T_{turtle1\_world} * T_{world\_turtle2}$$

可视化工具



$ rosrun rviz rviz -d `rospack find turtle_tf` /rviz/turtle_rviz.rviz

2. TF坐标变换

```cpp
/**
 * 该例程产生tf数据，并计算、发布turtle2的速度指令
 */

#include <ros/ros.h>
#include <tf/transform_broadcaster.h>
#include <turtlesim/Pose.h>

std::string turtle_name;

void poseCallback(const turtlesim::PoseConstPtr& msg)
{
    // 创建tf的广播器
    static tf::TransformBroadcaster br;

    // 初始化tf数据
    tf::Transform transform;
    transform.setOrigin( tf::Vector3(msg->x, msg->y, 0.0) );
    tf::Quaternion q;
    q.setRPY(0, 0, msg->theta);
    transform.setRotation(q);

    // 广播world与海龟坐标系之间的tf数据
    br.sendTransform(tf::StampedTransform(transform, ros::Time::now(), "world", turtle_name));
}

int main(int argc, char** argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "my_tf_broadcaster");

    // 输入参数作为海龟的名字
    if (argc != 2)
    {
        ROS_ERROR("need turtle name as argument");
        return -1;
    }

    turtle_name = argv[1];

    // 订阅海龟的位姿话题
    ros::NodeHandle node;
    ros::Subscriber sub = node.subscribe(turtle_name+"/pose", 10, &poseCallback);

    // 循环等待回调函数
    ros::spin();

    return 0;
};
```

turtle_tf_broadcaster.cpp

如何实现一个tf广播器

- 定义TF广播器（TransformBroadcaster）
- 创建坐标变换值；
- 发布坐标变换（sendTransform）

2. TF坐标变换

```cpp
int main(int argc, char** argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "my_tf_listener");

    // 创建节点句柄
    ros::NodeHandle node;

    // 请求产生turtle2
    ros::service::waitForService("/spawn");
    ros::ServiceClient add_turtle = node.serviceClient<turtlesim::Spawn>("/spawn");
    turtlesim::Spawn srv;
    add_turtle.call(srv);

    // 创建发布turtle2速度控制指令的发布者
    ros::Publisher turtle_vel = node.advertise<geometry_msgs::Twist>("/turtle2/cmd_vel", 10);

    // 创建tf的监听器
    tf::TransformListener listener;

    ros::Rate rate(10.0);
    while (node.ok())
    {
        // 获取turtle1与turtle2坐标系之间的tf数据
        tf::StampedTransform transform;
        try
        {
            listener.waitForTransform("/turtle2", "/turtle1", ros::Time(0), ros::Duration(3.0));
            listener.lookupTransform("/turtle2", "/turtle1", ros::Time(0), transform);
        }
        catch (tf::TransformException &ex)
        {
            ROS_ERROR("%s",ex.what());
            ros::Duration(1.0).sleep();
            continue;
        }

        // 根据turtle1与turtle2坐标系之间的位置关系，发布turtle2的速度控制指令
        geometry_msgs::Twist vel_msg;
        vel_msg.angular.z = 4.0 * atan2(transform.getOrigin().y(),
                                        transform.getOrigin().x());
        vel_msg.linear.x = 0.5 * sqrt(pow(transform.getOrigin().x(), 2) +
                                      pow(transform.getOrigin().y(), 2));
        turtle_vel.publish(vel_msg);

        rate.sleep();
    }
    return 0;
};
```

turtle_tf_listener.cpp

如何实现一个TF监听器

- 定义TF监听器；

（TransformListener）

- 查找坐标变换；

（waitForTransform、 lookupTransform）

2. TF坐标变换

```
## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)
target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})

add_executable(turtle_tf_listener src/turtle_tf_listener.cpp)
target_link_libraries(turtle_tf_listener ${catkin_LIBRARIES})
```

如何配置CMakeLists.txt中的编译规则

- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；

```
add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)
target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})

add_executable(turtle_tf_listener src/turtle_tf_listener.cpp)
target_link_libraries(turtle_tf_listener ${catkin_LIBRARIES})
```

CMakeLists.txt

2. TF坐标变换

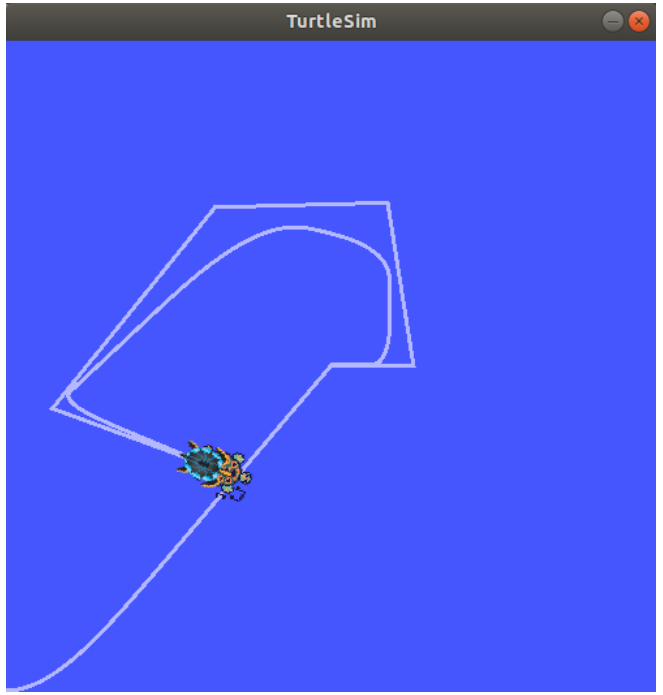
```
<launch>

   <!-- Turtlesim Node-->
   <node pkg="turtlesim" type="turtlesim_node" name="sim"/>
   <node pkg="turtlesim" type="turtle_teleop_key" name="teleop" output="screen"/>

   <node pkg="learning_tf" type="turtle_tf_broadcaster" args="/turtle1" name="turtle1_tf_broadcaster" />
   <node pkg="learning_tf" type="turtle_tf_broadcaster" args="/turtle2" name="turtle2_tf_broadcaster" />

   <node pkg="learning_tf" type="turtle_tf_listener" name="listener" />

 </launch>
```

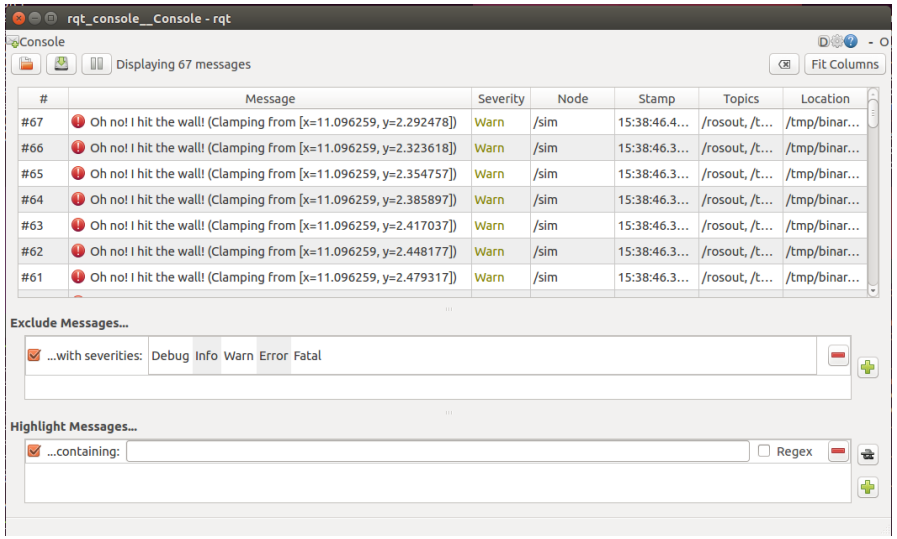


运行海龟
跟随例程

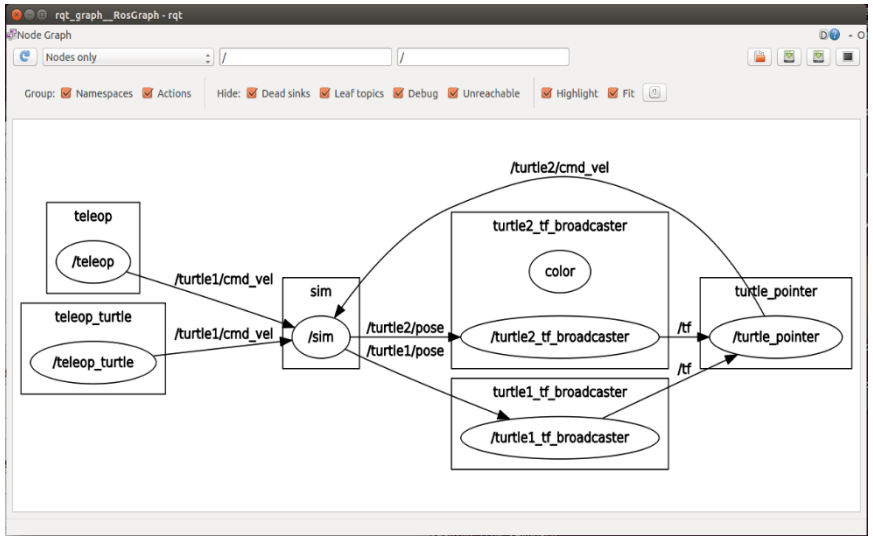
$ roslaunch learning_tf start_tf_demo_c++.launch

3. 可视化显示与仿真工具

3. 可视化显示与仿真工具

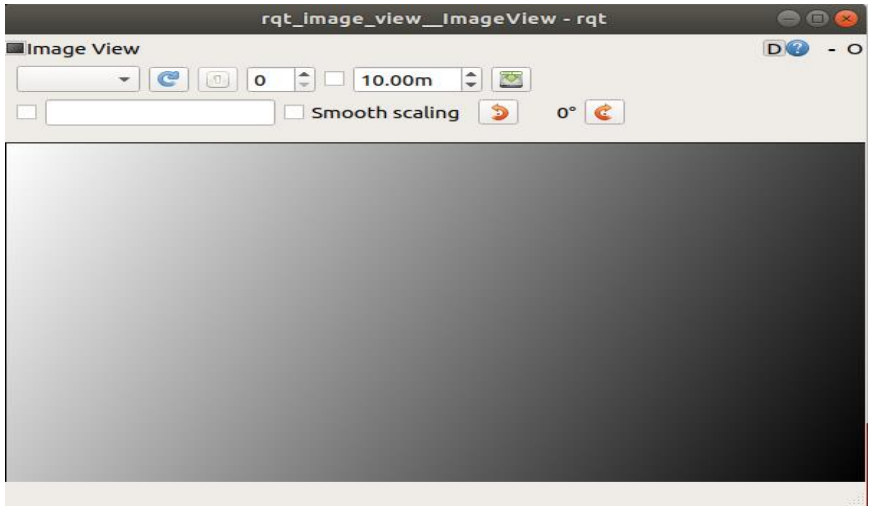


日志输出工具——rqt_console



计算图可视化工具——rqt_graph

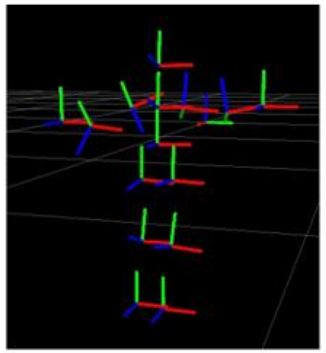
数据绘图工具——rqt_plot



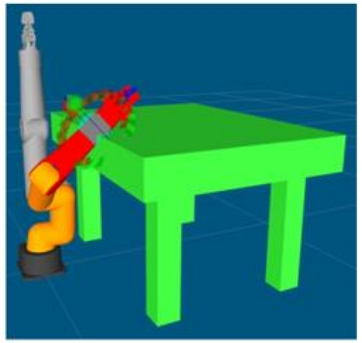
图像渲染工具——rqt_image_view

3. 可视化显示与仿真工具

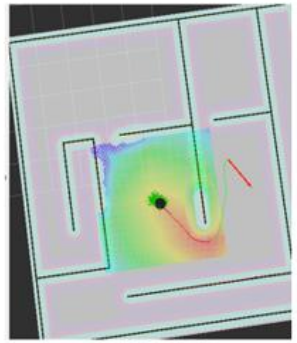
机器人模型



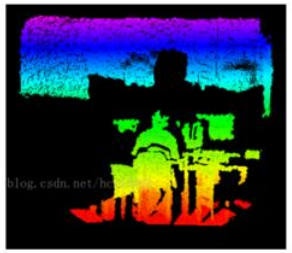
坐标



运动规划

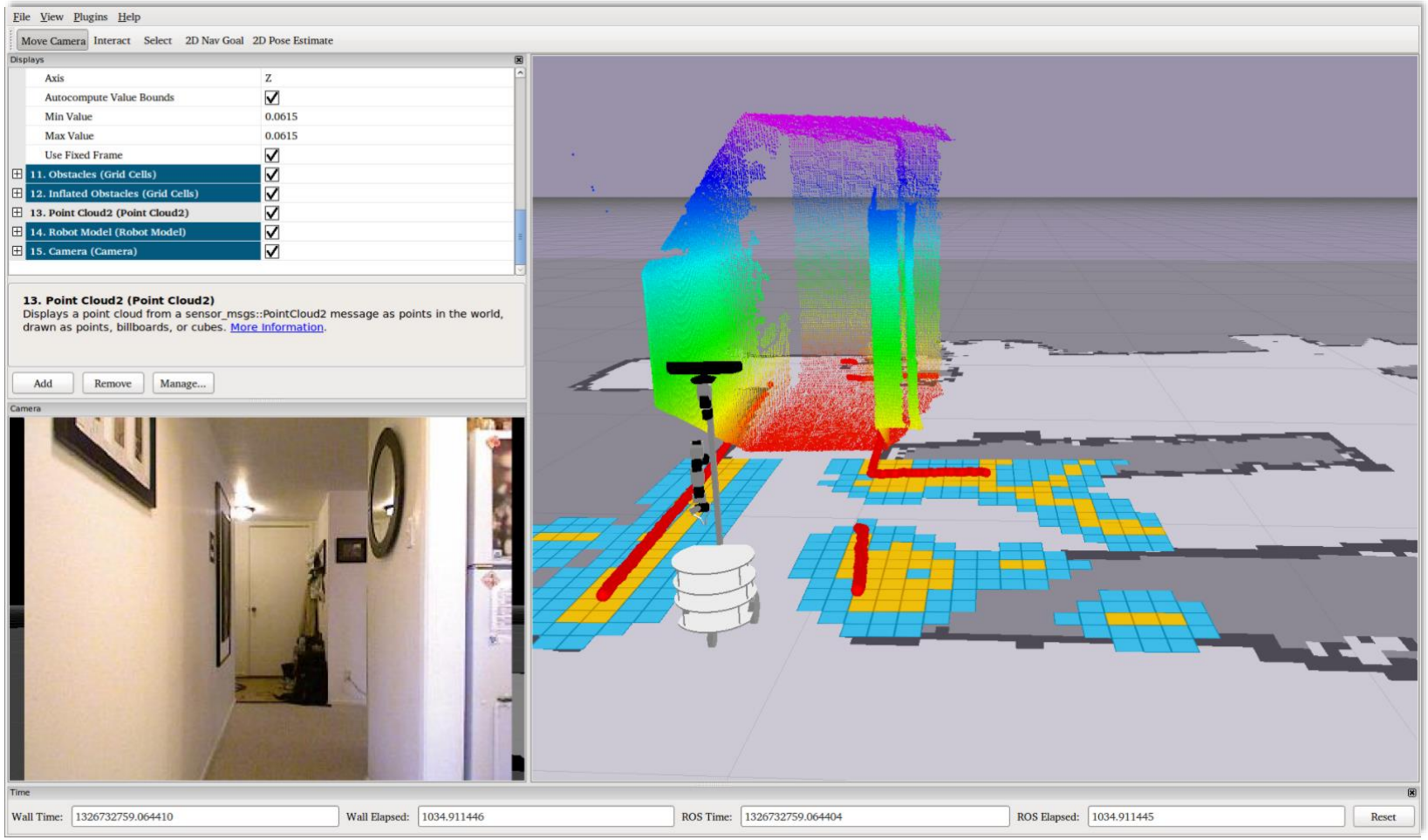


导航



点云

图像



机器人开发过程中的数据可视化界面

3. 可视化显示与仿真工具

古月居 GYH.AI

Rviz是一款**三维可视化工具**，可以很好的兼容基于ROS软件框架的机器人平台。

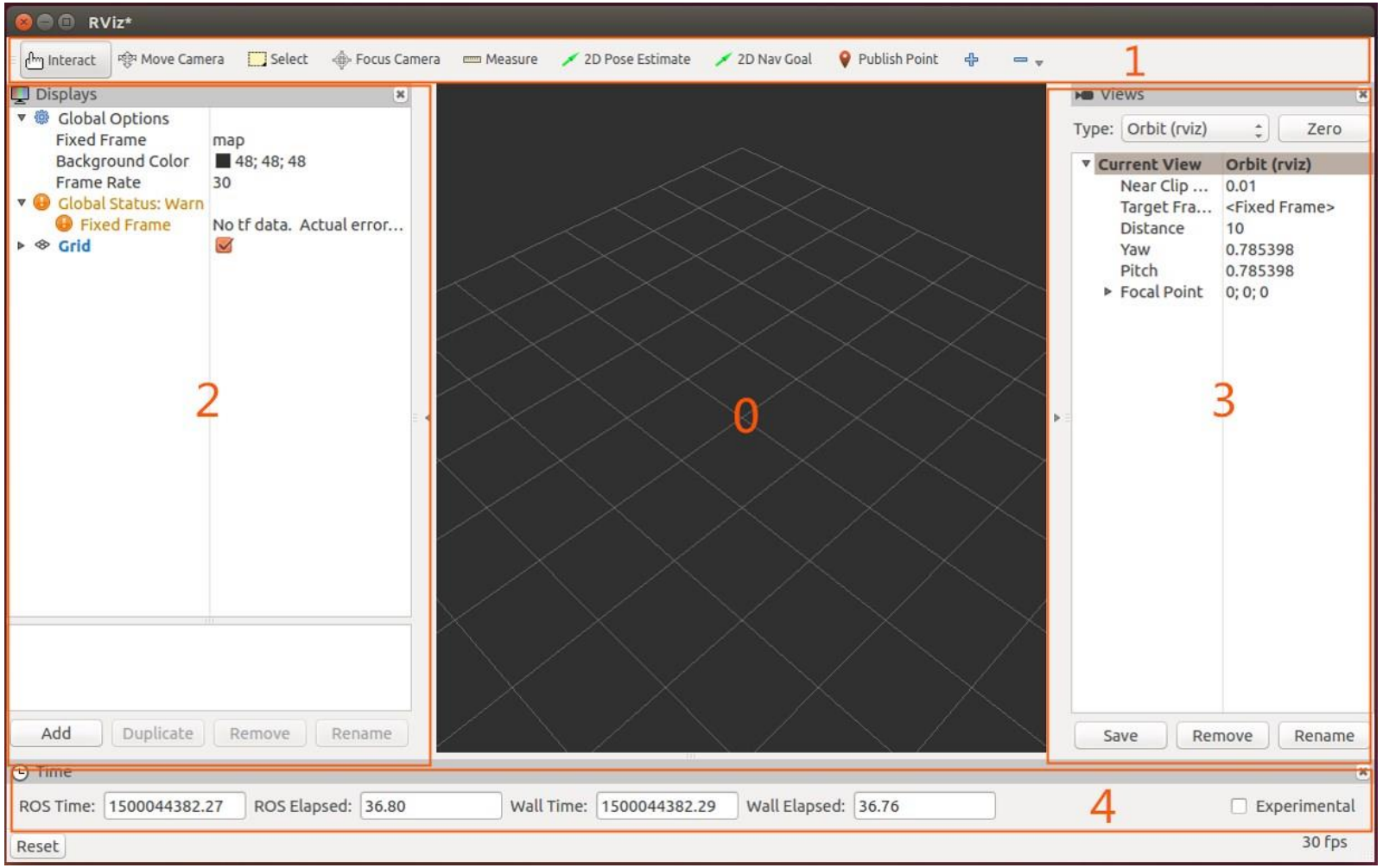
- 在rviz中，可以使用**可扩展标记语言XML**对机器人、周围物体等任何实物进行尺寸、质量、位置、材质、关节等属性的描述，并且在界面中呈现出来。

- 同时，rviz还可以通过**图形化的方式**，实时显示机器人传感器的信息、机器人的运动状态、周围环境的变化等信息。

- 总而言之，rviz通过机器人模型参数、机器人发布的传感信息等数据，为用户进行所有**可监测信息的图形化显示**。用户和开发者也可以在rviz的控制界面下，通过按钮、滑动条、数值等方式，控制机器人的行为。

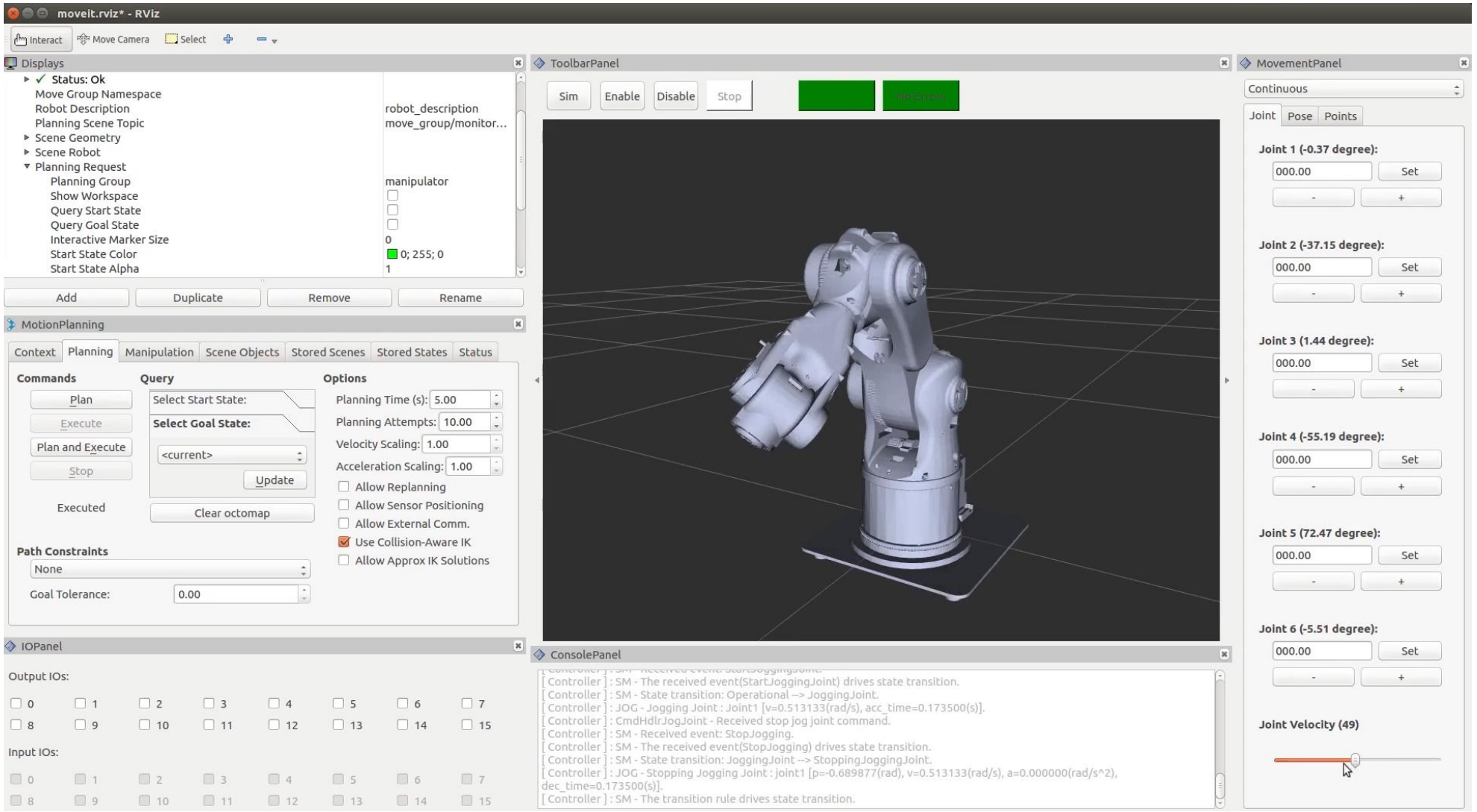
3. 可视化显示与仿真工具

- 0： 3D视图区
- 1： 工具栏
- 2： 显示项列表
- 3： 视角设置区
- 4： 时间显示区



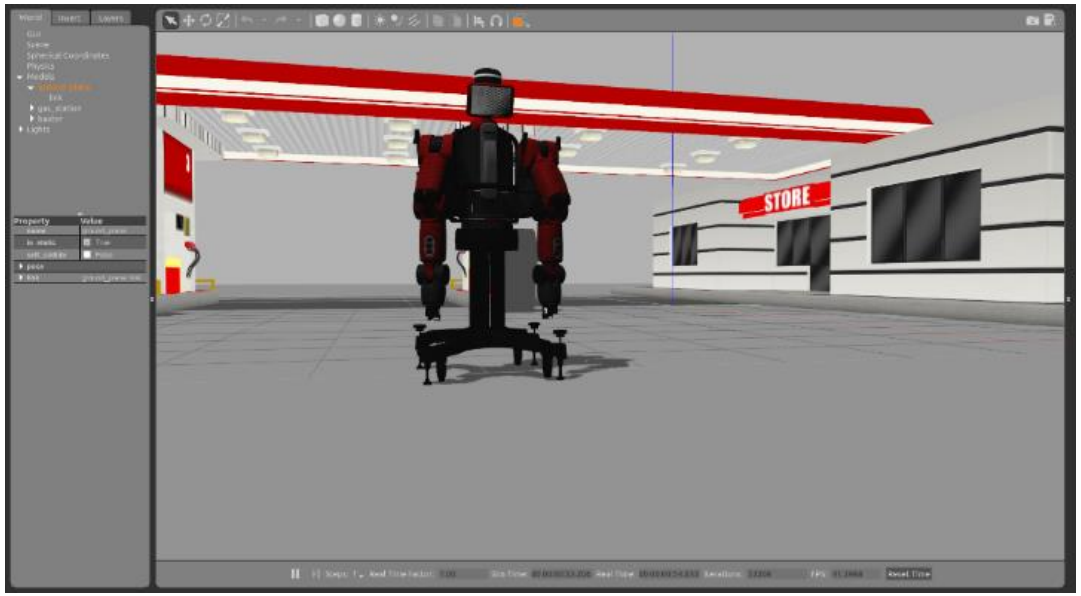
$ rosrun rviz rviz

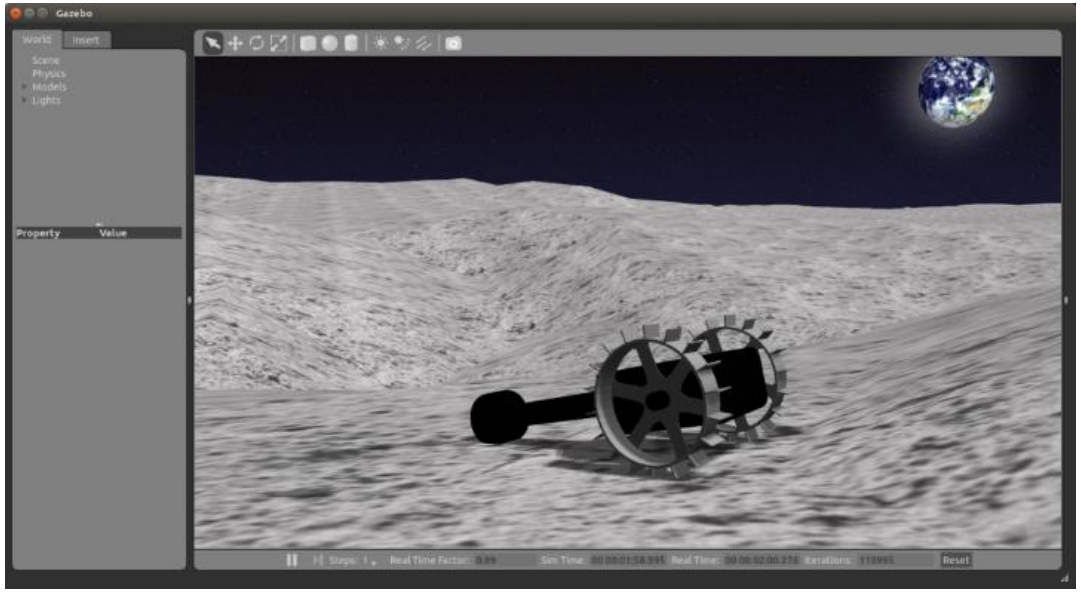
3. 可视化显示与仿真工具

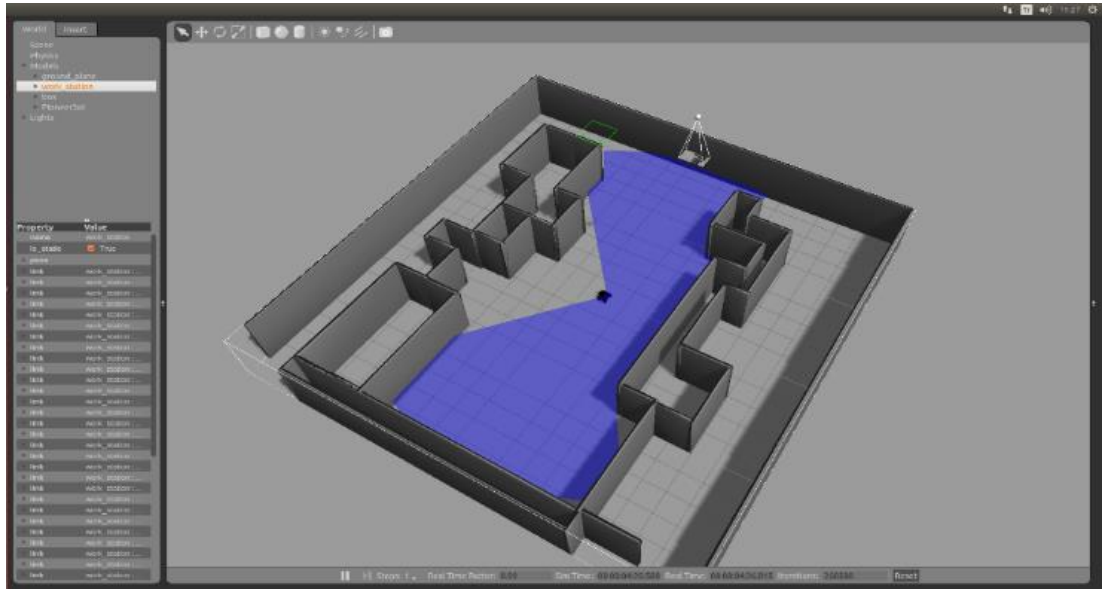


基于Rviz打造自己的人机交互界面

3. 可视化显示与仿真工具

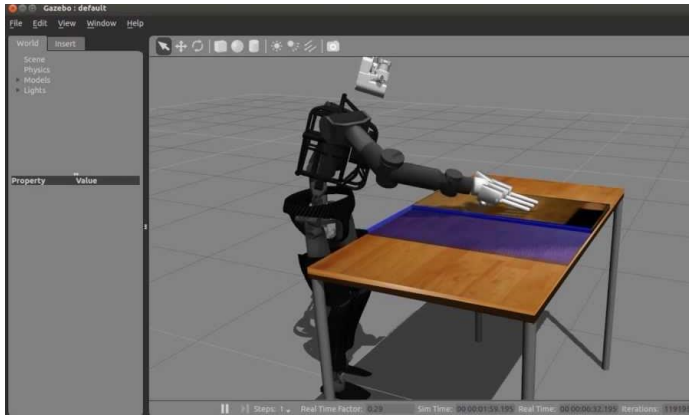






3. 可视化显示与仿真工具

- Gazebo是一个三维动态物理仿真器，能准确高效地仿真在复杂的室内外环境下机器人群体。与游戏引擎类似，Gazebo能对一整套传感器进行高度逼真的物理仿真、为程序和用户提供交互接口，其典型应用场景包括：
  - 测试机器人算法
  - 机器人的设计
  - 现实情景下的回溯测试



Features

Dynamics Simulation

Access multiple high-performance physics engines including ODE, Bullet, Simbody, and DART.

Advanced 3D Graphics

Utilizing OGRE, Gazebo provides realistic rendering of environments including high-quality lighting, shadows, and textures.

Sensors and Noise

Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras, Kinect style sensors, contact sensors, force-torque, and more.

Plugins

Develop custom plugins for robot, sensor, and environmental control. Plugins provide direct access to Gazebo's API.

Robot Models

Many robots are provided including PR2, Pioneer2 DX, iRobot Create, and TurtleBot. Or build your own using SDF.

TCP/IP Transport

Run simulation on remote servers, and interface to Gazebo through socket-based message passing using Google Protobufs.

Cloud Simulation

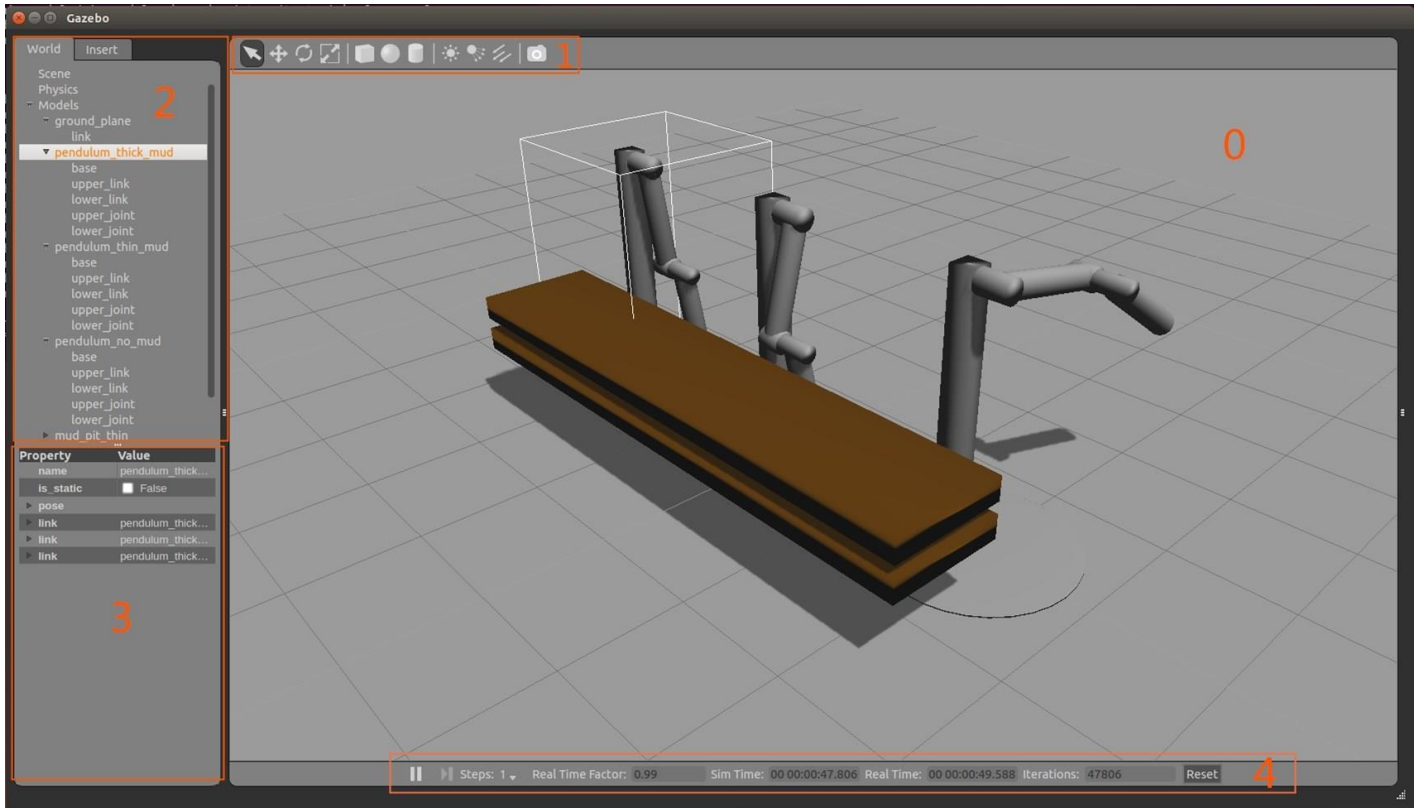
Use CloudSim to run Gazebo on Amazon AWS and GzWeb to interact with the simulation through a browser.

Command Line Tools

Extensive command line tools facilitate simulation introspection and control.

3. 可视化显示与仿真工具

- 0： 3D视图区
- 1： 工具栏
- 2： 模型列表
- 3： 模型属性项
- 4： 时间显示区



$ roslaunch gazebo_ros mud_world.launch

建议：为保证模型顺利加载，请提前将模型文件库下载并放置到~/.gazebo/models下
https://bitbucket.org/osrf/gazebo_models/downloads/

3. 可视化显示与仿真工具

古月居 GYH.AI

配置机器人模型

创建仿真环境

开始仿真

如何使用Gazebo进行仿真

本讲小结

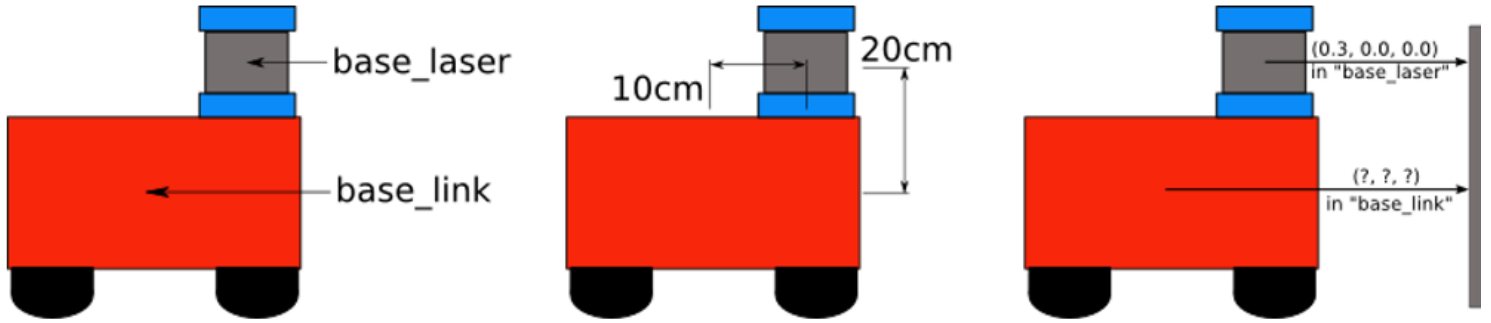
Launch启动文件	<launch>、<node>、<param>、<rosparam> <arg>、<include>、<remap>
TF坐标变换	TF工具的使用 广播TF变换、监听TF变换
可视化显示与仿真工具	rqt、rviz、gazebo

本讲作业

1. 创建一个learning_launch功能包，在其中新建launch文件，分别完成第3讲三道题目的启动和测试，将每道题目中使用的所有rosrun命令替换为一个roslaunch命令。

2. 下载gazebo离线模型库，并放置在指定位置，成功运行gazebo后，在界面中添加模型进行测试。

3. 创建一个learning_tf功能包，完成tf的编程和测试：已知激光雷达和机器人底盘的坐标关系，广播并监听机器人的坐标变换，求解激光雷达数据在底盘坐标系下的坐标值：



扩展阅读

- 古月·ROS入门21讲
https://www.bilibili.com/video/av59458869

- ROS Launch
http://wiki.ros.org/roslaunch/XML

- ROS技术点滴 —— launch文件
https://mp.weixin.qq.com/s/qY_NpuEiKl5cDH0NexyP5g

- ROS技术点滴 —— 海龟例程中的tf
https://mp.weixin.qq.com/s/O930feF67v7uxhlwmSFh9w

- ROS探索总结（二十二）——设置机器人的tf变换
http://www.guyuehome.com/355

- 《Introduction to Robotics Mechanics and Control》
《机器人学导论》，第二章

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭