ROS

ITMO more than a UNIVERSITY

Операционная система ROS

Разработка программных решений для роботов

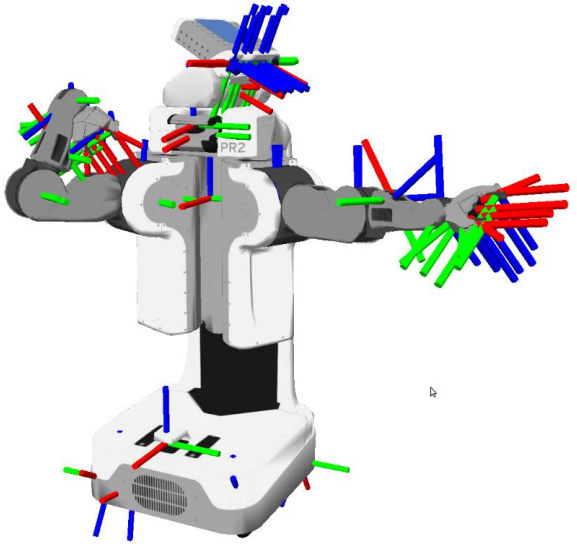
План занятия

- Опрос на15 мин
- Система преобразований TF
- RQT user interface
- Обзор Rviz
- Создание модели робота (URDF)
  - URDF
  - Xacro
- Robot state publisher & Joint state publisher
- 10 мин на вопросы по лабораторным

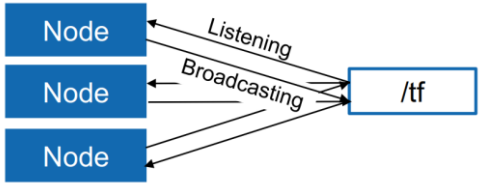
Система преобразований TF

Чтобы описывать движение объектов в пространстве удобно бывает прикрепить к ним относительные системы координат. А переход из одной СК в другую задавать преобразованием.

- Инструмент для отслеживания систем координат в течении времени
- Поддерживает взаимосвязь между рамками координат в древовидной структуре, буферизованной во времени.
- Позволяет пользователю преобразовывать точки, векторы и т. д. между системами координат в нужное время.
- Реализован как publisher/subscriber в топиках **/tf** и **/tf_static**



Используется специальный тип сообщений, у которого есть “header”.



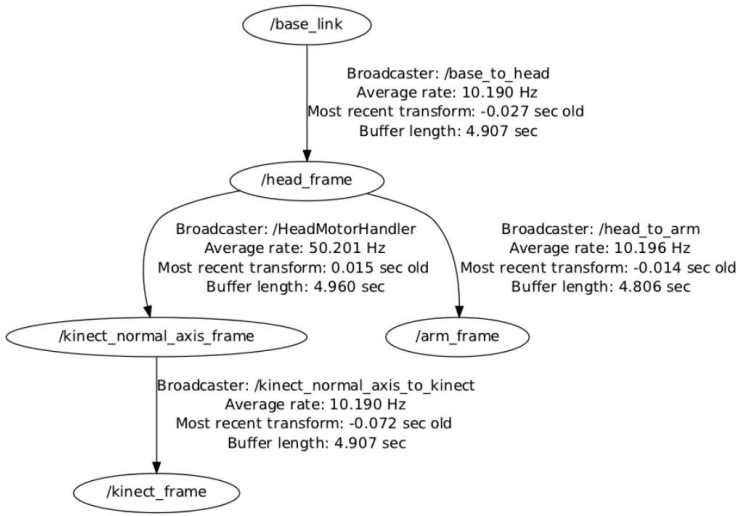
Ссылка на вики: http://ros.org/wiki/tf2

Transform Tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```

rosrun tf view_frames



Можно запрашивать преобразования между СК в любой момент времени.
См. tf_listener

rosrun tf view_frames – вывод дерева преобразований
evince frames.pdf – просмотр файла

TF tools

Command line

Вывод информации о текущем дереве преобразований

rosrun tf tf_monitor

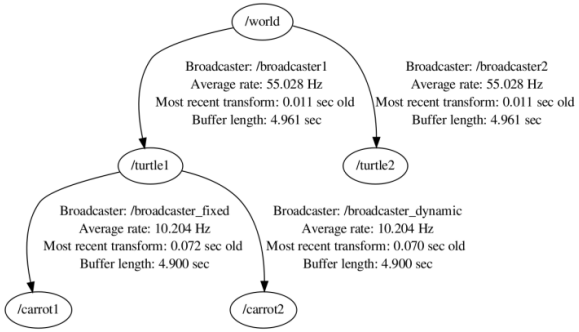
Вывод информации о преобразовании между двумя СК

rosrun tf tf_echo source_frame target_frame

View Frames

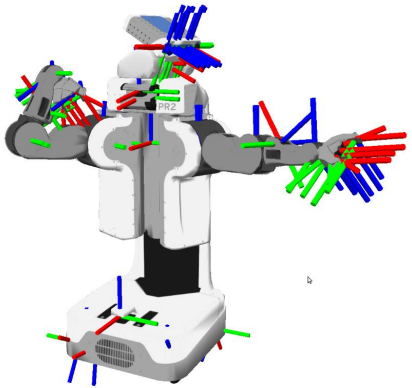
Создает визуальный граф дерева преобразований (PDF)

rosrun tf view_frames



Rviz

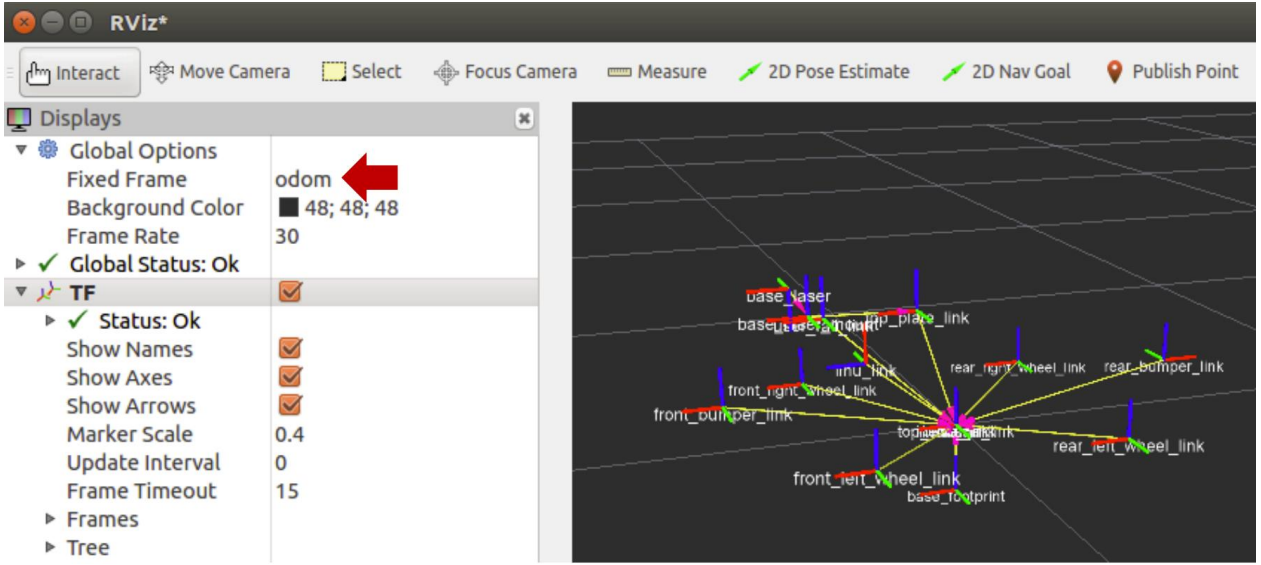
Трехмерная визуализация преобразований



RViz Plugin

Требуется настройка базовой системы координат (фиксированной) + добавить данные для визуализации

Настройки Rviz можно сохранить и позже открыть.



Запуск:
rosrun rviz rviz

Ссылка на вики: http://wiki.ros.org/rviz

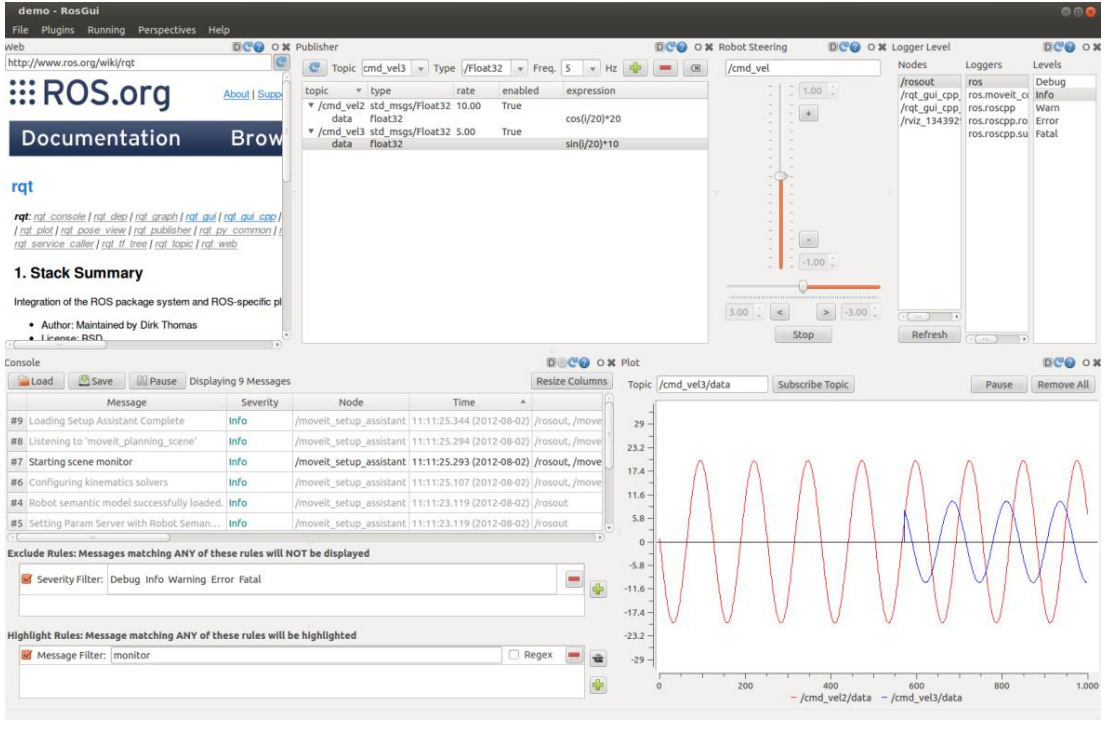
rqt User Interface

- Пользовательский интерфейс на основе Qt
- Пользовательские интерфейсы могут быть настроены
- Существует множество готовых плагинов
- Простота написания собственных плагинов

rosrun rqt_gui rqt_gui

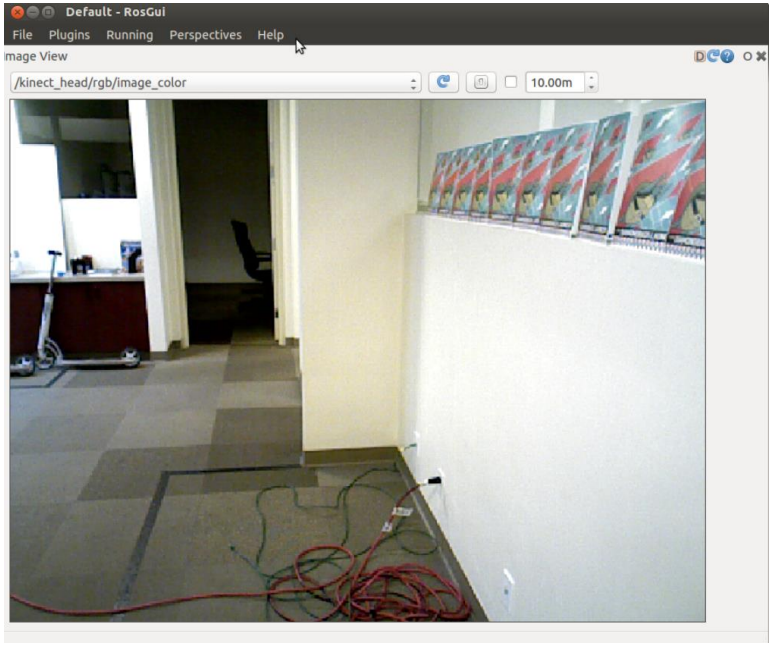
или rqt

Ссылка на вики:
http://wiki.ros.org/rqt/Plugins



Примеры плагинов

Визуализатор изображений rqt_image_view



Прочие:

- rqt_graph
- rqt_multiplot
- rqt_console
- rqt_logger_level

Ссылка на wiki: http://wiki.ros.org/rqt_image_view

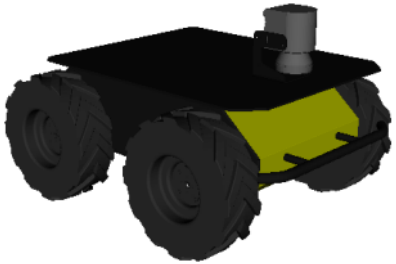
URDF - Unified Robot Description Format

Задает в формате XML описание модели робота:

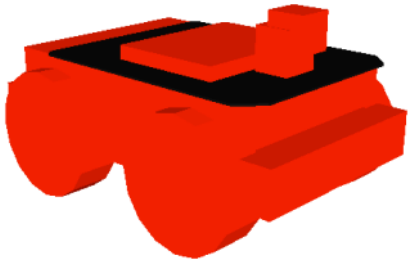
- Описывает кинематику и динамику
- Визуальное представление
- Модель столкновения
- URDF может формироваться при помощи макросов

Рассмотрим на примере!

Геометрия для визуализации



Примитивы для столкновения

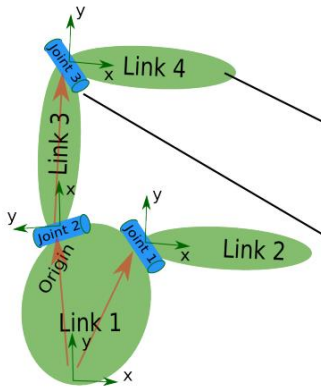


Ссылка на wiki: http://wiki.ros.org/urdf

URDF пример

- Описание состоит из набора элементов (звеньев) и набора соединительных элементов (шарниров).
- Шарниры соединяют звенья.

У каждого элемента (Link) может быть 3 свойства: Visual, Colission, Intertia.
У каждого шарнира (Joint) может быть: Parent, child



robot.urdf

```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
    </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" …/>
  </inertial>
</link>
```

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" … />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

Ссылка на вики: http://wiki.ros.org/urdf/xml

URDF tags

Типы шарниров:

- **revolute** — шарнирное соединение, которое вращается вдоль оси и имеет ограниченный диапазон, определяемый верхним и нижним пределами.
- **continuous** — непрерывное шарнирное соединение, вращающееся вокруг оси и не имеющее верхних и нижних пределов.
- **prismatic** — скользящее соединение, которое скользит вдоль оси и имеет ограниченный диапазон, определяемый верхним и нижним пределами.
- **fixed** — На самом деле это не шарнир, потому что он не может двигаться. Все степени свободы заблокированы. Этот тип соединения не требует **<axis>**, **<calibration>**, **<dynamics>**, **<limits>** или **<safety_controller>**.
- **floating** — допускает движение по всем 6 степеням свободы.
- **planar** — допускает движение в плоскости, перпендикулярной оси.

Список тегов:

robot Описывает все свойства робота.
Link Описывает кинематические и динамические свойства звена.
transmission Трансмиссии связывают приводы с шарнирами и представляют собой их механическое соединение.
joint Описывает кинематические и динамические свойства шарнира
gazebo Описывает параметры моделирования, такие как демпфирование, трение и т. д.
sensor Описывает датчик, например камеру, лучевой датчик и т. д.
model_state Описывает состояние модели в определенный момент времени.
model Описывает кинематические и динамические свойства конструкции робота.

Модель робота обычно хранится в robot_description сервера параметров и его можно визуализировать в Rviz при помощи плагина Robot Model

XACRO

XML Macros (macro language)

С помощью xacro вы можете создавать более короткие и более читаемые XML-файлы, расширяют возможности xml-разметки.

Пример использования в launch

```
<xacro:property name="the_radius" value="2.1" />
<xacro:property name="the_length" value="4.5" />

<geometry type="cylinder" radius="${the_radius}" length="${the_length}" />
```

```
<xacro:property name="radius" value="4.3" />
<circle diameter="${2 * radius}" />
```

```
<xacro:if value="<expression>">
  <... some xml code here ...>
</xacro:if>
```

```
<xacro:arg name="myvar" default="false"/>
```

```
<xacro:include filename="other_file.xacro" ns="namespace"/>
```

Ссылка на вики: http://wiki.ros.org/xacro

SDF – Simulation Description Format

Определяет формат XML для описания:

- Окружающей среды (освещение, гравитация и т. д.)
- Объекты (статические и динамические)
- Датчики
- Роботы

SDF — стандартный формат для Gazebo

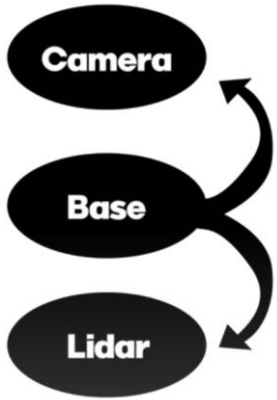
Gazebo преобразует URDF в SDF автоматически

Ссылка на вики: http://sdformat.org/spec

robot_state_publisher

Этот пакет позволяет публиковать состояние робота в tf2. Как только состояние будет опубликовано, оно станет доступно всем компонентам системы, которые также используют tf2. Пакет принимает углы сочленений робота в качестве входных данных и публикует трехмерные положения звеньев робота, используя кинематическую древовидную модель робота.
Использовать в качестве узла ROS (не как библиотеку).

ROS API:
joint_states (sensor_msgs/JointState) – информация о положении шарнира
ROS Param:
robot_description (urdf map)
tf_prefix (string)
publish_frequency (double)




```
<launch>
 <!-- Load the urdf into the parameter server. -->
 <param name="my_robot_description" textfile="$(find mypackage)/urdf/robotmodel.xml"/>

 <node pkg="robot_state_publisher" type="robot_state_publisher" name="rob_st_pub" >
    <remap from="robot_description" to="my_robot_description" />
    <remap from="joint_states" to="different_joint_states" />
  </node>
</launch>
```

http://wiki.ros.org/robot_state_publisher/Tutorials/Using%20the%20robot%20state%20publisher%20on%20your%20own%20robot

joint_state_publisher

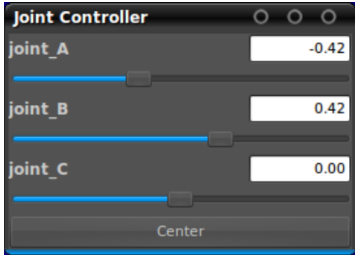
Этот пакет содержит инструмент для настройки и публикации значений состояния шарниров из заданного URDF.

- Считывает параметр robot_description с сервера параметров, находит все незафиксированные шарниры и публикует сообщение JointState со всеми этими шарнирами.
- Публикует сообщения Sensor_msgs/JointState для робота.

ROS API:
- joint_states (sensor_msgs/JointState)
- robot_description (String, default: None)
- dependent_joints (Map of Maps, default: None)
- rate (Integer, default: 10 Hz)

```
<node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher"/>
```

2 интерфейса:
- ROS API
- Графический интерфейс GUI



Ссылка на вики: http://wiki.ros.org/joint_state_publisher

ITMO re than a UNIVERSITY

Спасибо за внимание!