Практика Gazebo ROS

Первая симуляция робота

1. Создадим новый пакет и назовем его **my_robot_simulation**

   `catkin_create_pkg my_robot_simulation rospy std_msgs`

2. Создадим внутри пакета файл запуска

   `mkdir launch`

со следующим содержанием

> 💡 **Всегда при копировании файлов или фрагментов кода(или разметки) проверяйте корректность указанных путей (включая название файлов).**

```
<launch>
<arg name="rviz_conf_file" default="$(find my_robot_model)/urdf/rviz_conf.rviz" />

<param name="robot_description" command="xacro '$(find my_robot_model)/urdf/diff_robot_model_done.urdf.xacro'" />

<!-- <param name="robot_description" command="$(find xacro)/xacro.py '$(find my_robot_model)/models/diff_drive_robot/urdf/diff_drive_robot.

<!-- <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" /> -->
<node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui" type="joint_state_publisher_gui" />

<node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher" output="screen" >
  <param name="publish_frequency" type="double" value="40.0" />
</node>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find my_robot_model)/worlds/empty.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>

  <node name="my_robot_model_spawn" pkg="gazebo_ros" type="spawn_model" output="screen"
   args="-urdf -param robot_description -model my_robot_model" />


<node name="rviz" pkg="rviz" type="rviz" args="-d $(arg rviz_conf_file)" />
</launch>
```

> 💡 Если вы отредактировали файл запуска, может потребоваться перезапуск roscore (имейте это ввиду, если что-то начнет работать не так, как ожидалось, и ошибок тоже не обнаружено).

3. Для работы симулятора обязательно должны быть теги **colission** и **inertial (не нулевые массы!!!)** у всех звенье, поэтому в соответствующи файлах исправьте это.

Далее пример разметки для звена base_link

```
<link name="base_link">
    <pose>0 0 0.2 0 0 0</pose>
    <visual>
      <origin xyz="0.0 0.0 0" rpy="0.0 0.0 0.0"/>
      <geometry>
          <box size="0.6 0.6 0.2"/>
      </geometry>
      <material name="green" />
    </visual>
```

```
    <collision>
      <origin xyz="0.0 0.0 0." rpy="0.0 0.0 0.0"/>
      <geometry>
          <box size="0.6 0.6 0.2"/>
      </geometry>
    </collision>
    <inertial>
      <origin xyz="0 0 0." rpy="0 0 0"/>
      <mass value="0.1"/>
      <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
    </inertial>
  </link>
```

5. Правильная настройка инерционных характристик материала является важной задачей, от которого будет зависть качество симуляции и сейчас мы с вами реализуем способ, который упростит работу. Всего существует 3 типа примитивов:
    - цилиндр
    - кубойд
    - сфера

инерционные характристики зависят только от формы и плотности(только масса), поэтому мы можем написать макросы, которыми можно пользоваться.

Файл с макросами можно скачать с репозитория, папка **practice 4/urdf/inertial.usdf.xacro**

Далее приведен пример использования макроса (не забудьте подключить макрос перед использованием).

```
<link name="left_wheel">
    <xacro:inertial_cylinder mass="0.1" length="0.05" radius="0.05">
        <origin xyz="0 0 0" rpy="0 0 0"/>
    </xacro:inertial_cylinder>
</link>
```

6. Теперь необходимо указать важные свойства для работы симулятора, начнем с указания цветов для газебо. Создадим файл **robot_diff.gazebo** и пропишем внутри

```
<?xml version="1.0"?>
<robot>
  <!-- SET COLORS FOR GAZEBO -->
  <gazebo reference="base_link">
      <material>Gazebo/Green</material>
  </gazebo>
  <gazebo reference="demo_link">
      <material>Gazebo/Red</material>
  </gazebo>



</robot>
```

Чтобы мы могли управлять роботом в симляторе, нам понадобится подключить плагин, сделаем это в том же файле **robot_diff.gazebo** путем добавления следующего блока

```
  <gazebo>
    <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
      <legacyMode>false</legacyMode>
      <alwaysOn>true</alwaysOn>
      <updateRate>30</updateRate>
      <leftJoint>wheel_left_joint</leftJoint>
      <rightJoint>wheel_right_joint</rightJoint>
      <wheelSeparation>0.3</wheelSeparation>
      <wheelDiameter>0.21</wheelDiameter>
      <wheelTorque>20</wheelTorque>
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>odom</odometryTopic>
```

```
      <odometryFrame>odom</odometryFrame>
      <robotBaseFrame>base_link</robotBaseFrame>
    </plugin>
  </gazebo>
```

Убедитесь, что параметры плагина были настроены корректно!

7. Теперь необходимо создать наш базовый файл мира **.world**

```
<sdf version="1.4">
  <world name="default">

    <scene>
      <ambient>0.4 0.4 0.4 1</ambient>
      <background>0.7 0.7 0.7 1</background>
      <shadows>true</shadows>
    </scene>

    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>

    <physics type="ode">
      <real_time_update_rate>1000.0</real_time_update_rate>
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1</real_time_factor>
      <ode>
        <solver>
          <type>quick</type>
          <iters>150</iters>
          <precon_iters>0</precon_iters>
          <sor>1.400000</sor>
          <use_dynamic_moi_rescaling>1</use_dynamic_moi_rescaling>
        </solver>
        <constraints>
          <cfm>0.00001</cfm>
          <erp>0.2</erp>
          <contact_max_correcting_vel>2000.000000</contact_max_correcting_vel>
          <contact_surface_layer>0.01000</contact_surface_layer>
        </constraints>
      </ode>
    </physics>
  </world>

</sdf>
```

Если все сделано правильно, то после запуска launch файла у вас запустится симулятор.

8. Теперь мы можем запустить симулятор и проверить корректность работы.

`roslaunch launch/robot_sim.launch`

однако управлять роботом при помощи **robot_joint_state_publisher** у вас не получится.

Интерфейс для управления роботом предоставляет плагин, который мы подключили. Соответственно, после запуска симуляции у нас появится топик **/cmd_vel** в который мы и можем отсылать команды для управления.

9. У нашего робота всего 2 колеса, таким роботом сложно управлять, поэтому мы можес сделать ему третье колесо, только в этот раз другого типа. Третье колесо будет сферическим элементом, которое создает минимальное трение о поверхность пола. Обратите внимание, здесь выполняется упрощенная имитация пассивного сферического колеса.

   Добавим к описанию base_link следующую разметку

```
<collision name='caster_collision'>
      <origin xyz="0.0 -0.25 -0.11" rpy="0.0 0.0 0.0"/>
      <geometry>
        <sphere radius="0.1" />
      </geometry>
      <surface>
        <friction>
          <ode>
            <mu>0</mu>
            <mu2>0</mu2>
            <slip1>1.0</slip1>
            <slip2>1.0</slip2>
          </ode>
        </friction>
      </surface>
    </collision>
    <visual name='caster_visual'>
      <origin xyz="0.0 -0.25 -0.11" rpy="0.0 0.0 0.0"/>
      <geometry>
        <sphere radius="0.1" />
      </geometry>
      <material name="blue"/>
    </visual>
```

10. Рассмотрим более подробно самые важные свойства моделирования.

    Моделирование трения в шарнирах (жесткость и демпфирование). Также очень важно установить ограничения на момент и скорость в шарнирах!

```
<limit effort="100" velocity="100"/>
<joint_properties damping="0.0" friction="0.0"/>
```

11. Добавление объектов окружения из библиотеки объектов Gazebo

После чего необходимо сохранить сцену

12. Установите пакет для управления платформой в симуляторе

teleop_twist_keyboard

Или опубликуйте соответствующие сообщения в топик /**cmd_vel**

Задание

1. Создать сцену с объектами окружения (стандартные из Газебо), сохранить в файл с новым названием и настроить его подгрузку в файле запуска.
2. Написать скрипт, который будет отправлять сообщения в /cmd_vel (любые сообщения)
3. При выполении спавн необходимо задать координаты x y z как аргументы

Ссылки на ROS Wiki:

https://classic.gazebosim.org/tutorials?tut=build_robot&cat=build_robot

https://classic.gazebosim.org/tutorials?cat=guided_b&tut=guided_b2

https://classic.gazebosim.org/tutorials?tut=ros_gzplugins#Tutorial:UsingGazebopluginswithROS (Плагины gazebo)