BASIC UGV ROBOT MODELLING ON GAZEBO

Erencan Bulut

Creating a Workspace

- Before starting, check the ROS version and follow the instructions. The tutorial was created for *melodic* ROS version.

- **To check the ROS version:**

$ rosversion -d

- **Start with one of two ways:**

1. Use the Terminal (but the code always has to be written in this way):
   $ source /opt/ros/melodic/setup.bash

2. Use the *.bashrc* (only one time):
   - Open the *.bashrc* on the Terminal:
     $ gedit ~/.bashrc
   - Copy and paste the following code at the end of the file:
     source /opt/ros/melodic/setup.bash
   - Run the following code on the Terminal or restart the Terminal:
     $ bash

- To check the ROS environment:

$ printenv | grep ROS

- **Create a *workspace*:**

$ mkdir -p ~/catkin_ws/src
$ cd catkin_ws/
$ catkin_make

- Open the *.bashrc* on the Terminal:
  $ gedit ~/.bashrc
- Copy and paste the following code at the end of the file:
  source /home/pcadmin/catkin_ws/devel/setup.bash
- Run the following code on the Terminal or restart the Terminal:
  $ bash

- For the confirmation:

$ echo $ROS_PACKAGE_PATH

P.S.: If a program/code runs as using a *Terminal*, open another *Terminal* to write other codes there.

TurtleBot 3 Installation

- **Run the following codes on the Terminal to install the *TurtleBot3*:**

$ cd catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git -b melodic-devel
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git -b melodic-devel
$ cd ..
$ catkin_make

- **There are 2 ways to use *Wafflepi* model:**

    1. Use the Terminal (but the code always has to be written in this way):
       $ export TURTLEBOT3_MODEL=waffle_pi

    2. Use the *.bashrc* (only one time):
        - Open the *.bashrc* on the Terminal:
          $ gedit ~/.bashrc
        - Copy and paste the following code at the end of the file:
          export TURTLEBOT3_MODEL=waffle_pi
        - Run the following code on the Terminal or restart the Terminal:
          $ bash

- **To run the *TurtleBot3* in an empty world:**

$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch

- **To move the *TurtleBot3*:**

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

- **Following codes can be run if necessary to see:**

$ rqt_graph
$ rostopic list
$ rosnode list

- **To run the *rviz* node:**

$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch

- **To download and check TF Tree:**

$ sudo apt-get install ros-melodic-tf2-tools
$ rosrun tf2_tools view_frames.py

It will create a *.pdf* file in the home file. It looks like the graph on the right side

- **To run different simulation models on the Gazebo:**
- There are several *worlds* in the *catkin_ws/src/turtlebot3_sumulations/turtlebot3_gazebo/worlds*
- These worlds can be run as follows:

$ roslaunch turtlebot3_gazebo turtlebot3_autorace.launch

Creating a Simulation Environment on Gazebo

- **Run the Gazebo:**

$ gazebo

- Now, click Edit/Building Editor (or Ctrl+B) on the Gazebo
- Wall, door, windows etc. can be edited.
- After creating the environment, save it as *maze_environment* in home/pcadmin/building_editor_models
- The environment can be found in the *Insert* section on the Gazebo
- In the *insert* section, some tools can be edited such as *construction cone*
- After arranging the environment, save it as *maze_environment.world* in home/pcadmin
- *.world* file also can be edited by opening it with a *Text Editor* as follows:

```
<model name='Construction Cone_0'>
  <link name='link'>
    <collision name='collision'>       Collision is for the physical scale.
      <geometry>
        <mesh>
          <scale>5 5 5</scale>
          <uri>model://construction_cone/meshes/construction_cone.dae</uri>
        </mesh>
      </geometry>
      <max_contacts>10</max_contacts>
      <surface>
        <contact>
          <ode/>
        </contact>
        <bounce/>
        <friction>
          <torsional>
            <ode/>
          </torsional>
          <ode/>
        </friction>
      </surface>
    </collision>
    <visual name='visual'>  Visual is for what we see on the screen.
      <geometry>
        <mesh>
          <scale>5 5 5</scale>
          <uri>model://construction_cone/meshes/construction_cone.dae</uri>
        </mesh>
      </geometry>
    </visual>
```

- Construction Cone's scales that is created on the Gazebo can be changed with Text Editor.
- Collision is for the physical scale.
- Visual is for what we see on the screen.

Creating a Launch File for the Robot and Environment

- **Create a package:**

$ cd catkin_ws/src/
$ catkin_create_pkg maze_environment
$ cd ..
$ catkin_make
$ cd src/maze_environment/
$ mkdir worlds
$ cp maze_environment.world /home/pcadmin/catkin_ws/src/maze_environment/worlds

- **Create/Edit a *.launch* file :**

$ roscd maze_environment
$ mkdir launch

- Instead to cread a new *.launch* file…:
- Copy *turtlebot3_empty_world.launch* (home/pcadmin/catkin_ws/src/turtlebot3 _simulations/turtlebot3_gazebo/launch).
- Paste it in the *launch* folder (home/pcadmin/catkin_ws/src/maze_environment).
- Rename it as *maze_environement.launch*
- Open it with a *Text Editor*
- Rewrite the 8th row as:
  <arg name="world_name" value="$(find maze_environment)/worlds/maze_environment.world"/>

```
1 <launch>
2   <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
3   <arg name="x_pos" default="0.0"/>
4   <arg name="y_pos" default="0.0"/>
5   <arg name="z_pos" default="0.0"/>
6
7   <include file="$(find gazebo_ros)/launch/empty_world.launch">
8     <arg name="world_name" value="$(find turtlebot3_gazebo)/worlds/empty.world"/>
9     <arg name="paused" value="false"/>
10    <arg name="use_sim_time" value="true"/>
11    <arg name="gui" value="true"/>
12    <arg name="headless" value="false"/>
13    <arg name="debug" value="false"/>
14  </include>
15
16  <param name="robot_description" command="$(find xacro)/xacro --inorder $(find turtlebot3_description)/urdf/turtlebot3_$(arg model).urdf.xacro" />
17
18  <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf" args="-urdf -model turtlebot3_$(arg model) -x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -param robot_description" />
19 </launch>
```

- **To run the new world/environment with *TurtleBot3*:**

$ roslaunch maze_environment maze_environment.launch

- **To move the *TurtleBot3*:**

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

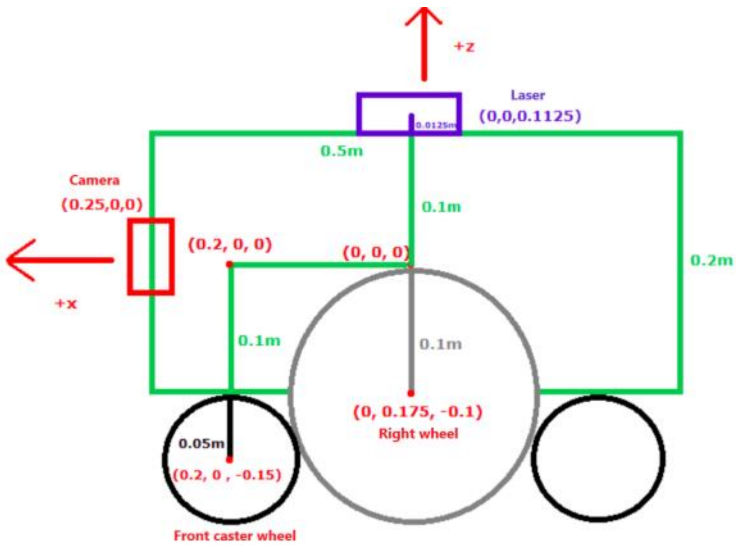
Robot Modeling in a Simulation Environment

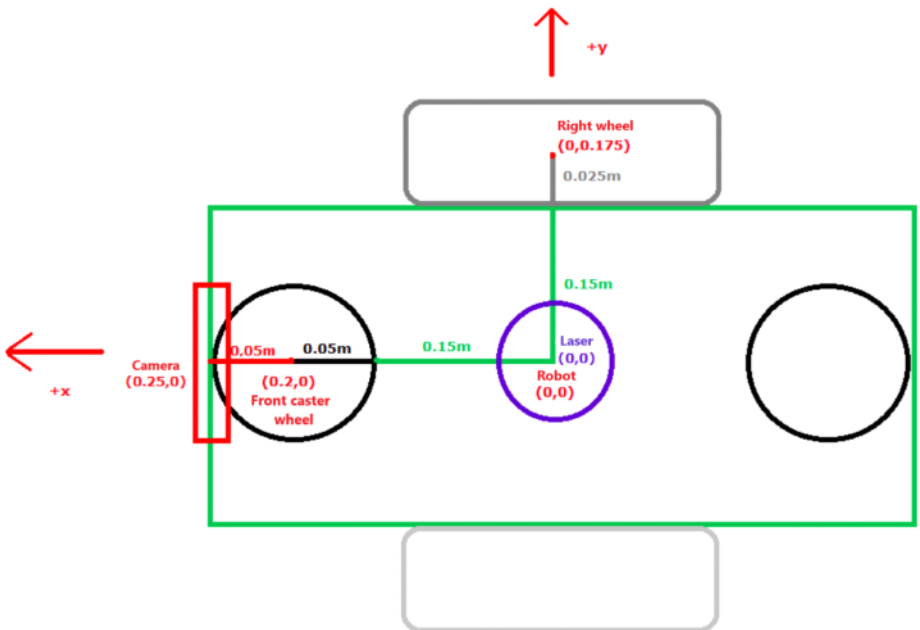
Features of the Robot:

- Differential Drive
- 2x Wheel
- 2x caster wheel
- Laser sensor (360°)
- RGB-D camera

Packages:

1) Description Package
    - Robot design
2) Gazebo Package
    - Launch files





- **Create the following packages:**

$ cd ~/catkin_ws/src
$ catkin_create_pkg robot_description
$ catkin_create_pkg robot_gazebo

- **Create the *urdf* and *launch* files:**

$ cd robot_description
$ mkdir urdf
$ cd ..
$ cd robot_gazebo
$ mkdir launch
$ cd ~/catkin_ws/
$ catkin_make

robot_base.xacro

- **Create the *robot_base.xacro* file:**

```
$ roscd robot_description/urdf
$ gedit robot_base.xacro
```

- **After creating the *.xacro* file (*Plain Text* should be chosen as *.XML*), the definitions should be made:**

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:macro name="robot_base">
   <link name="base_footprint">
    <collision>   The collision is defined.
     <origin xyz="0 0 0" rpy="0 0 0"/>
     <geometry>
       <box size="0.5 0.3 0.2"/>   The box size (0.5x0.3x0.2 m³) is defined.
     </geometry>
    </collision>
    <visual>   The visual is defined.
     <origin xyz="0 0 0" rpy="0 0 0"/>
     <geometry>
       <box size="0.5 0.3 0.2"/>
     </geometry>
    </visual>
    <xacro:box_inertial x="0.5" y="0.3" z="0.2" mass="10.0"/>   The inertia of the box is defined here. An inertia file
                                                                 will be imported to calculate the inertia of the box.
   </link>
   <gazebo reference="base_footprint">   The definition for the Gazebo
    <material value="Gazebo/Green"/>   The Color is defined as green
   </gazebo>
 </xacro:macro>
</robot>
```

- **Since all the *.xacro* files will be stored in a *.xacro* file, the *robot_body.xacro* file is created:**

```
$ roscd robot_description/urdf
$ gedit robot_body.xacro
```

- **Definitions of the *robot_body.xacro:***

```
<?xml version="1.0"?>
<robot name="robot" xmlns:xacro="http://ros.org/wiki/xacro">   Robot is named "Robot" here.
    <xacro:include filename="$(find robot_description)/urdf/robot_base.xacro"/>   In order to ´Import´ the
                                                                                  robot_base.xacro file,
                                                                                  it is addressed.
   <xacro:robot_base/>
 </robot>
```

- **The *robot_body.xacro* file should be edited after creating each of *.xacro* file. The last version of *the robot_body.xacro* file will be shown in the last page.**

robot_inertia.xacro

http://wiki.ros.org/urdf/Tutorials/Adding%20Physical%20and%20Collision%20Properties%20to%20a%20URDF%20Model

The inertia file is used for other the other inertia definitions.

- **Create the *robot_inertia.xacro* file:**

$ roscd robot_description/urdf
$ gedit robot_inertia.xacro

- **Copy the codes in the following link and paste them into the *robot_inertia.xacro* file:**

https://github.com/uos/uos_tools/blob/fuerte/uos_common_urdf/common.xacro

- **Add the inertia part into the *robot_body.xacro:***

```
<?xml version="1.0"?>
<robot name="robot" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:include filename="$(find robot_description)/urdf/robot_base.xacro"/>
  <xacro:include filename="$(find robot_description)/urdf/robot_inertia.xacro"/>

  <xacro:robot_base />
</robot>
```

- **The *robot_body.xacro* file should be edited after creating each of *.xacro* file. The last version of *the robot_body.xacro* file will be shown in the last page.**

robot_wheel.xacro

- **Create the *robot_wheel.xacro* file:**

```
$ roscd robot_description/urdf
$ gedit robot_wheel.xacro
```

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:macro name="robot_wheel" params="xyz_coordinate rpy_coordinate direction">  It is named "robot_wheel". In order to define the
 <link name="wheel_${direction}_link">  Each link should have its own name!          fixed points of the wheels, params are defined.
  <collision>
    <origin xyz="0 0 0" rpy="0 1.570796 1.570796"/> Pitch (pi/2) and yaw (pi/2) angles (90) are defined in the radian unit.
    <geometry>
      <cylinder radius="0.10" length="0.05"/> The radius  (0.1 m) and the length (0.05 m) of the wheel are defined.
    </geometry>
  </collision>
  <visual> The same for the visual!
    <origin xyz="0 0 0" rpy="0 1.570796 1.570796"/>
    <geometry>
      <cylinder radius="0.10" length="0.05"/>
    </geometry>
  </visual>
  <xacro:cylinder_inertial radius="0.1" length="0.05" mass="0.5"/>  The inertia of the wheel (m = 0.5 kg) is defined.
 </link>

 <joint name="wheel_${direction}_base" type="continuous">  A´joint´ should be defined to show where the link is joined.
  <origin xyz="${xyz_coordinate}" rpy="${rpy_coordinate}"/>
  <parent link="base_footprint"/>
  <child link="wheel_${direction}_link"/>
  <axis xyz="0 1 0"/>  Since the wheel turns on the y-axis, it is defined as "0 1 0".
 </joint>

 <gazebo reference="wheel_${direction}_link">  The definition for the Gazebo
  <material value="Gazebo/Black"/> The color is defined as black
 </gazebo>

 </xacro:macro>

 <xacro:macro name="robot_caster_wheel" params="xyz_coordinate rpy_coordinate direction">  A caster wheel is defined. In order to define the
 <link name="caster_wheel_${direction}_link">                                               fixed points of the wheel, params are defined.
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <sphere radius="0.05"/>  Its shape is the sphere and the radius is 0.05 m.
    </geometry>
  </collision>
  <visual> The same for the visual!
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <sphere radius="0.05"/>
    </geometry>
  </visual>
  <xacro:sphere_inertial radius="0.05" mass="0.05"/> The inertia of the wheel (m = 0.05 kg; r = 0.05 m) is defined.
 </link>

 <joint name="caster_wheel_${direction}_base" type="continuous">  A´joint´ should be defined to show where the link is joined.
  <origin xyz="${xyz_coordinate}" rpy="${rpy_coordinate}"/>
  <parent link="base_footprint"/>  from where
  <child link="caster_wheel_${direction}_link"/>  to where
  <axis xyz="1 1 1"/>  Since the wheel turns on the x,y,z-axis, it is defined as "1 1 1".
 </joint>

 <gazebo reference="caster_wheel_${direction}_link"> The definition for the Gazebo
  <material value="Gazebo/White"/>  The color is defined as black
 </gazebo>

 </xacro:macro>

</robot>
```

robot_differential.xacro

- **Create the *robot_differential.xacro* file:**

$ roscd robot_description/urdf
$ gedit robot_differential.xacro

- **Copy the codes in the following link (Differential Drive) and paste them into the *robot_differential.xacro* file:**

https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

- **Then edit the codes according to the robot:** Explanations are written above the codes.

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:macro name="robot_differential">  The macro name is written.
       <gazebo>
        <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">

         <!-- Plugin update rate in Hz -->
         <updateRate>10.0</updateRate>

         <!-- Name of left joint, defaults to `left_joint` -->
         <leftJoint>wheel_left_base</leftJoint>

         <!-- Name of right joint, defaults to `right_joint` -->
         <rightJoint>wheel_right_base</rightJoint>

         <!-- The distance from the center of one wheel to the other, in meters, defaults to 0.34 m -->
         <wheelSeparation>0.35</wheelSeparation>

         <!-- Diameter of the wheels, in meters, defaults to 0.15 m -->
         <wheelDiameter>0.2</wheelDiameter>

         <!-- Wheel acceleration, in rad/s^2, defaults to 0.0 rad/s^2 -->
         <wheelAcceleration>0.0</wheelAcceleration>

         <!-- Maximum torque which the wheels can produce, in Nm, defaults to 5 Nm -->
         <wheelTorque>5</wheelTorque>

         <!-- Topic to receive geometry_msgs/Twist message commands, defaults to `cmd_vel` -->
         <commandTopic>cmd_vel</commandTopic>

         <!-- Topic to publish nav_msgs/Odometry messages, defaults to `odom` -->
         <odometryTopic>odom</odometryTopic>

         <!-- Odometry frame, defaults to `odom` -->
         <odometryFrame>odom</odometryFrame>

         <!-- Robot frame to calculate odometry from, defaults to `base_footprint` -->
         <robotBaseFrame>base_footprint</robotBaseFrame>

         <!-- Odometry source, 0 for ENCODER, 1 for WORLD, defaults to WORLD -->
         <odometrySource>1</odometrySource>

         <!-- Set to true to publish transforms for the wheel links, defaults to false -->
         <publishWheelTF>false</publishWheelTF>

         <!-- Set to true to publish transforms for the odometry, defaults to true -->
         <publishOdom>true</publishOdom>

         <!-- Set to true to publish sensor_msgs/JointState on /joint_states for the wheel joints, defaults to false -->
         <publishWheelJointState>false</publishWheelJointState>

         <!-- Set to true to swap right and left wheels, defaults to true -->
         <legacyMode>true</legacyMode>
        </plugin>
       </gazebo>
 </xacro:macro>
</robot>
```

robot_laser.xacro

- **Create the *robot_laser.xacro* file:**

$ roscd robot_description/urdf
$ gedit robot_laser.xacro

- The codes in the following link (under the "rrbot.gazebo, again as we did for the camera example:") are used: https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:macro name="robot_laser">
   <link name="laser_link">
     <collision>
       <origin xyz="0 0 0" rpy="0 0 0"/>  The coordinate of the lidar is defined.
       <geometry>
         <cylinder radius="0.025" length="0.025"/>  The size of the lidar is defined (r = 0.025 m; l = 0.025 m).
       </geometry>
     </collision>
     <visual>  The same for the visual.
       <origin xyz="0 0 0" rpy="0 0 0"/>
       <geometry>
         <cylinder radius="0.025" length="0.025"/>
       </geometry>
     </visual>
     <xacro:cylinder_inertial radius="0.025" length="0.025" mass="0.2"/>  The inertia of the lidar (m = 0.2 kg) is defined.
   </link>
   <joint name="base_to_laser" type="fixed">
     <origin xyz="0.0 0.0 0.1125" rpy="0 0 0"/>  Since it is fixed on the `base`, z-axis is "0.01125" and rpy is "0 0 0".
     <parent link="base_footprint"/> from where
     <child link="laser_link"/>  to where
   </joint>
  <gazebo reference="laser_link"> The definition for the Gazebo
   <material value="Gazebo/Blue"/>  The color is defined as blue
   <sensor type="gpu_ray" name="head_hokuyo_sensor">   The codes under this row are imported.
     <pose>0 0 0 0 0 0</pose>
     <visualize>false</visualize>  For the observation, it can be changed as "true"
     <update_rate>40</update_rate>
     <ray>
       <scan>
         <horizontal>
           <samples>720</samples>
           <resolution>1</resolution>
           <min_angle>-3.1415</min_angle>  Scanning from -90 degree to +90 degree is defined: min: -1.570796 max: +1.570796 (in radius).
           <max_angle>3.1415</max_angle>  Scanning from -180 degree to +180 degree is defined: min: -3.1415 max: +3.1415 (in radius).
         </horizontal>
       </scan>
       <range>
         <min>0.10</min>
         <max>10.0</max>
         <resolution>0.01</resolution>
       </range>
       <noise>
         <type>gaussian</type>
         <!-- Noise parameters based on published spec for Hokuyo laser
             achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and
             stddev of 0.01m will put 99.7% of samples within 0.03m of the true
             reading. -->
         <mean>0.0</mean>
         <stddev>0.01</stddev>
       </noise>
     </ray>
     <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_gpu_laser.so">
       <topicName>scan</topicName>  The topic name is changed.
       <frameName>laser_link</frameName>  The frame name is changed.
     </plugin>
   </sensor>
 </gazebo>
 </xacro:macro>
</robot>
```

robot_camera.xacro

$ roscd robot_description/urdf
$ gedit robot_camera.xacro

- The codes in the following link (under the "Openni Kinect") are used: https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:macro name="robot_camera">
   <link name="camera_link">
     <collision>
       <origin xyz="0 0 0" rpy="0 0 0"/> The coordinate of the camera is defined.
       <geometry>
         <box size="0.025 0.1 0.025"/>  The size of the camera is defined (x = 0.025 m; y = 0.1 m; z = 0.025 m).
       </geometry>
     </collision>
     <visual>  The same for the visual
       <origin xyz="0 0 0" rpy="0 0 0"/>
       <geometry>
         <box size="0.025 0.1 0.025"/>
       </geometry>
     </visual>
     <xacro:box_inertial x="0.025" y="0.1" z="0.025" mass="0.1"/>  The inertia of the camera (m = 0.1 kg) is defined.
   </link>
   <joint name="base_to_camera" type="fixed"> A´joint´ should be defined to show where the link is joined.
     <origin xyz="0.25 0 0" rpy="0 0 0"/> Since it is fixed on the `base`, x-axis is "0.25" and rpy is "0 0 0".
     <parent link="base_footprint"/>  from where
     <child link="camera_link"/>  to where
   </joint>

           <gazebo reference="camera_link">  The definition for the Gazebo
            <material value="Gazebo/Red"/>  The color is defined as red
The codes under      <sensor name="camera_link_camera" type="depth"> The name is changed to "camera_link_camera"
this row are  imported.  <update_rate>20</update_rate>
             <camera>
              <horizontal_fov>1.047198</horizontal_fov>
              <image>
                   <width>640</width>
                   <height>480</height>
                   <format>R8G8B8</format>
              </image>
              <clip>
                   <near>0.05</near>
                   <far>3</far>
              </clip>
             </camera>
             <plugin name="camera_link_controller" filename="libgazebo_ros_openni_kinect.so">  The name is changed to "camera_link_camera"
              <baseline>0.2</baseline>
              <alwaysOn>true</alwaysOn>
              <updateRate>1.0</updateRate>
              <cameraName></cameraName>
              <imageTopicName>/camera/rgb/image_raw</imageTopicName>
              <cameraInfoTopicName>/camera/rgb/camera_info</cameraInfoTopicName>              These names are
              <depthImageTopicName>/camera/depth/image_raw</depthImageTopicName>              changed as on the
              <depthImageInfoTopicName>/camera/depth/camera_info</depthImageInfoTopicName>    left.
              <pointCloudTopicName>/camera/depth/points</pointCloudTopicName>
              <frameName>camera_link</frameName>
              <pointCloudCutoff>0.5</pointCloudCutoff>
              <pointCloudCutoffMax>3.0</pointCloudCutoffMax>
              <distortionK1>0.00000001</distortionK1>
              <distortionK2>0.00000001</distortionK2>
              <distortionK3>0.00000001</distortionK3>
              <distortionT1>0.00000001</distortionT1>
              <distortionT2>0.00000001</distortionT2>
              <CxPrime>0</CxPrime>
              <Cx>0</Cx>
              <Cy>0</Cy>
              <focalLength>0</focalLength>
              <hackBaseline>0</hackBaseline>
             </plugin>
            </sensor>
           </gazebo>

 </xacro:macro>
</robot>
```

robot_body.xacro

- **Last version of the *robot_body.xacro* file should be as following:**

```
$ roscd robot_description/urdf
$ gedit robot_body.xacro
```

```
<?xml version="1.0"?>
<robot name="robot" xmlns:xacro="http://ros.org/wiki/xacro">
    <xacro:include filename="$(find robot_description)/urdf/robot_base.xacro"/>
    <xacro:include filename="$(find robot_description)/urdf/robot_inertia.xacro"/>
    <xacro:include filename="$(find robot_description)/urdf/robot_wheel.xacro"/>
    <xacro:include filename="$(find robot_description)/urdf/robot_differential.xacro"/>
    <xacro:include filename="$(find robot_description)/urdf/robot_laser.xacro"/>
    <xacro:include filename="$(find robot_description)/urdf/robot_camera.xacro"/>

    <xacro:robot_base />
```

For the wheels, coordinates are defined according to the robot shame in page 6.

```
    <xacro:robot_wheel xyz_coordinate="0.0 -0.175 -0.1" rpy_coordinate="0 0 0" direction="left"/>
    <xacro:robot_wheel xyz_coordinate="0.0 0.175 -0.1" rpy_coordinate="0 0 0" direction="right"/>
    <xacro:robot_caster_wheel xyz_coordinate="0.2 0.0 -0.15" rpy_coordinate="0 0 0"
    direction="front"/>
    <xacro:robot_caster_wheel xyz_coordinate="-0.2 0.0 -0.15" rpy_coordinate="0 0 0"
    direction="back"/>
    <xacro:robot_differential />
    <xacro:robot_laser />
    <xacro:robot_camera />

</robot>
```

robot_gazebo.launch

- **Create the *robot_gazebo.lauch* file:**

$ roscd robot_gazebo/launch
$ gedit robot_gazebo.launch

```
<?xml version="1.0"?>
<launch>
  <arg name="robot_coordinate" default="-x 0.0 -y 0.0 -z 0.00 -R 0.0 -P 0.0 -Y 0.0" />  Starting point of the robot
  <arg name="robot_name" default="/" />
  <include file="$(find gazebo_ros)/launch/empty_world.launch">  The world which the robot is started.
    <arg name="world_name" value="/worlds/empty.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>
    <param name="robot_description" command="$(find xacro)/xacro '$(find
robot_description)/urdf/robot_body.xacro'"/>
    <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf" args="-urdf -model $(arg
robot_name) -param robot_description $(arg robot_coordinate) "/>
    <node pkg="robot_state_publisher" type="robot_state_publisher"
name="robot_state_publisher">
    </node>
    <node pkg="joint_state_publisher" type="joint_state_publisher"
name="joint_state_publisher">
    </node>
</launch>
```

- **To open the simulation:**

$ roslaunch robot_gazebo robot_gazebo.launch

- **To move the robot:**

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch