# Übungsblatt 2

## Aufgabe 1

In dieser Übung lernen wir, wie man einen Roboter mit **Gazebo** simulieren kann. Dafür schauen wir uns zunächst an, was Gazebo eigentlich ist.

**Was ist Gazebo?**

Gazebo ist eine 3D-Simulationsumgebung mit einer Physik-Engine. Sie erlaubt es, Roboter in einer realitätsnahen Umgebung zu simulieren — mit Schwerkraft, Kollisionen und Reibung. So kann man auch ohne echten Roboter entwickeln und testen.

Typische Vorteile:

- Entwicklung ohne reale Hardware
- Testen extremer Situationen
- Simulation schwer zugänglicher Umgebungen
- Visualisierung und Kontrolle vor dem Bau eines echten Roboters

**Gazebo vs. RViz**

RViz ist ein Visualisierungstool, das ROS-Daten wie `/tf`-Frames und Sensordaten anzeigt. Es simuliert nichts. Gazebo hingegen simuliert die Physik und kann Roboter realitätsnah verhalten lassen. Beides zusammen ergibt ein starkes Duo: Gazebo simuliert, RViz zeigt an.

### Installation und Start

Gazebo installieren

```
sudo apt install ros-jazzy-ros-gz
```

Gazebo starten:

```
gz sim empty.sdf
```

Alternativ über ROS 2:

```
ros2 launch ros_gz_sim gz_sim.launch.py gz_args:=empty.sdf
```

### Verbindung mit ROS 2

Gazebo und ROS 2 laufen getrennt, kommunizieren aber über `ros_gz_bridge`. Mit dieser Bridge können topics wie `/cmd_vel` oder `/joint_states` verbunden werden.

**Nächste Schritte**

1. URDF-Datei für Gazebo anpassen (`<inertial>` und `<collision>` hinzufügen)
2. Roboter in Gazebo spawnen
3. Plugins einbinden und ROS-Bridge konfigurieren

**Hinweis:** Die Simulation nutzt dieselbe ROS 2-Architektur wie ein echter Roboter — nur werden die Sensoren und Aktoren durch Gazebo-Plugins ersetzt.

a) HINWEIS: alle jetzt folgenden Aufgaben wurden bereits umgesetzt im dargestellten code. Hier wird nochmals veranschaulicht, was nötig ist.

b) Ok wir wissen was wir tun wollen. Wir arbeiten in dieser über mit dem Beispielcode der in: exercises/aktuell/uebung_2/code/ liegt. Zuerst kümmern wir uns darum eine valide urdf datei zu erzeugen, die auch alle nötigen Daten für Gazebo enthält. Dafür erweitern wir zuerst ein paar macros die uns helfen die inertial matrizen zu berechen. In der Datei common_properties.xacro ergänzen wir folgenden macros.

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

    <material name="green">
        <color rgba="0 0.6 0 1" />
    </material>

    <material name="gray">
        <color rgba="0.7 0.7 0.7 1" />
    </material>

    <xacro:macro name="box_inertia" params="m x y z o_xyz o_rpy">
        <inertial>
            <mass value="${m}" />
            <origin xyz="${o_xyz}" rpy="${o_rpy}" />
            <inertia ixx="${(m/12) * (z*z + y*y)}" ixy="0" ixz="0"
                     iyy="${(m/12) * (x*x + z*z)}" iyz="0"
                     izz="${(m/12) * (x*x + y*y)}" />
        </inertial>
    </xacro:macro>

    <xacro:macro name="cylinder_inertia" params="m r l o_xyz o_rpy">
        <inertial>
            <mass value="${m}" />
            <origin xyz="${o_xyz}" rpy="${o_rpy}" />
            <inertia ixx="${(m/12) * (3*r*r + l*l)}" ixy="0" ixz="0"
                     iyy="${(m/12) * (3*r*r + l*l)}" iyz="0"
                     izz="${(m/2) * (r*r)}" />
        </inertial>
    </xacro:macro>

    <xacro:macro name="sphere_inertia" params="m r o_xyz o_rpy">
        <inertial>
            <mass value="${m}" />
            <origin xyz="${o_xyz}" rpy="${o_rpy}" />
            <inertia ixx="${(2/5) * m * r * r}" ixy="0" ixz="0"
                     iyy="${(2/5) * m * r * r}" iyz="0"
                     izz="${(2/5) * m * r * r}" />
        </inertial>
    </xacro:macro>

</robot>
```

c) Danach passen wir usere mobile_base.xarco Datei an um nun die inertial tags und collision tags hinzuzufügen:

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

    <xacro:property name="base_length" value="0.6" />
    <xacro:property name="base_width" value="0.4" />
```

```
6     <xacro:property name="base_height" value="0.2" />
7     <xacro:property name="wheel_radius" value="0.1" />
8     <xacro:property name="wheel_length" value="0.05" />
9
10    <link name="base_footprint" />
11
12    <link name="base_link">
13        <visual>
14            <geometry>
15                <box size="${base_length} ${base_width} ${base_height}" />
16            </geometry>
17            <origin xyz="0 0 ${base_height / 2.0}" rpy="0 0 0" />
18            <material name="green" />
19        </visual>
20        <collision>
21            <geometry>
22                <box size="${base_length} ${base_width} ${base_height}" />
23            </geometry>
24            <origin xyz="0 0 ${base_height / 2.0}" rpy="0 0 0" />
25        </collision>
26        <xacro:box_inertia m="5.0" x="${base_length}" y="${base_width}" z="${base_height}"
27                           o_xyz="0 0 ${base_height / 2.0}" o_rpy="0 0 0" />
28    </link>
29
30    <xacro:macro name="wheel_link" params="prefix">
31        <link name="${prefix}_wheel_link">
32            <visual>
33                <geometry>
34                    <cylinder radius="${wheel_radius}" length="${wheel_length}" />
35                </geometry>
36                <origin xyz="0 0 0" rpy="${pi / 2.0} 0 0" />
37                <material name="gray" />
38            </visual>
39            <collision>
40                <geometry>
41                    <sphere radius="${wheel_radius}" />
42                </geometry>
43                <origin xyz="0 0 0" rpy="0 0 0" />
44            </collision>
45            <xacro:cylinder_inertia m="1.0" r="${wheel_radius}" l="${wheel_length}"
46                                    o_xyz="0 0 0" o_rpy="${pi / 2.0} 0 0" />
47        </link>
48    </xacro:macro>
49
50    <xacro:wheel_link prefix="right" />
51    <xacro:wheel_link prefix="left" />
52
53    <link name="caster_wheel_link">
54        <visual>
55            <geometry>
56                <sphere radius="${wheel_radius / 2.0}" />
57            </geometry>
58            <origin xyz="0 0 0" rpy="0 0 0" />
59            <material name="gray" />
60        </visual>
61        <collision>
62            <geometry>
63                <sphere radius="${wheel_radius / 2.0}" />
64            </geometry>
65            <origin xyz="0 0 0" rpy="0 0 0" />
66        </collision>
67        <xacro:sphere_inertia m="0.5" r="${wheel_radius / 2.0}"
68                              o_xyz="0 0 0" o_rpy="0 0 0" />
69    </link>
70
71    <joint name="base_joint" type="fixed">
72        <parent link="base_footprint" />
73        <child link="base_link" />
74        <origin xyz="0 0 ${wheel_radius}" rpy="0 0 0" />
75    </joint>
76
77    <joint name="base_right_wheel_joint" type="continuous">
78        <parent link="base_link" />
79        <child link="right_wheel_link" />
80        <origin xyz="${-base_length / 4.0} ${-(base_width + wheel_length) / 2.0} 0" rpy="0 0 0" />
81        <axis xyz="0 1 0" />
82    </joint>
83
84    <joint name="base_left_wheel_joint" type="continuous">
85        <parent link="base_link" />
86        <child link="left_wheel_link" />
87        <origin xyz="${-base_length / 4.0} ${(base_width + wheel_length) / 2.0} 0" rpy="0 0 0" />
```

```
88          <axis xyz="0 1 0" />
89      </joint>
90
91      <joint name="base_caster_wheel_joint" type="fixed">
92          <parent link="base_link" />
93          <child link="caster_wheel_link" />
94          <origin xyz="${base_length / 3.0} 0 ${-wheel_radius / 2.0}" rpy="0 0 0" />
95      </joint>
96
97  </robot>
```

d) Wir bauen noch eine gazebo simulationsdatei in der wir simulierte Motoren und Sensoren definieren können. Wir legen Sie auch im URDF Verzeichnis ab und inkludieren sie in der master urdf. Die neue Datei heißt, mobile_base_gazebo.xarco

e) Ich habe noch ein zusätzlichs ros_package ergänzt: my_robot_bringup, dort ist nur eine launch Datei enthalten und eine Konfiguration für Gazebo, wodurch wir dann Gazebo richtig initialisiert starten können.

f) Wir können nun alles starten über:

```
1 ros2 launch my_robot_bringup my_robot.launch.xml
```

g) Jetzt können wir versuchen den Roboter fahren zu lassen!

```
1 ros2 topic pub /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0
      .0, z: 1.0}}"
```