Mecanum Robot - Step 4: RViz Visualization

Real-time 3D Visualization with Odometry and IMU

Overview

This step adds real-time 3D visualization in RViz. The robot model moves according to odometry and IMU data published by the Portenta H7 via micro-ROS WiFi.

Features

- Simple robot model (box chassis + cylinder wheels)
- Real-time position and orientation from /odom
- TF broadcast from odometry data
- Trajectory visualization with Path display

Workspace Setup

```
cd ~
mkdir -p mecanum_viz_ws/src
cd mecanum_viz_ws/src
ros2 pkg create --build-type ament_python mecanum_description
cd mecanum_description
mkdir -p urdf launch rviz
```

File: urdf/mecanum_robot.urdf

```
<?xml version="1.0"?>
<robot name="mecanum_robot">

  <!-- Base Link -->
  <link name="base_link">
    <visual>
      <geometry>
        <box size="0.35 0.30 0.10"/>
      </geometry>
      <origin xyz="0 0 0.05" rpy="0 0 0"/>
      <material name="blue">
        <color rgba="0.2 0.4 0.8 1"/>
      </material>
    </visual>
  </link>

  <!-- IMU Link -->
  <link name="imu_link">
    <visual>
      <geometry>
        <box size="0.02 0.02 0.01"/>
      </geometry>
      <material name="red">
        <color rgba="1 0 0 1"/>
      </material>
    </visual>
  </link>

  <joint name="imu_joint" type="fixed">
    <parent link="base_link"/>
    <child link="imu_link"/>
    <origin xyz="0 0 0.1" rpy="0 0 0"/>
  </joint>
```

```
<!-- Front Left Wheel -->
<link name="fl_wheel">
  <visual>
    <geometry>
      <cylinder radius="0.048" length="0.04"/>
    </geometry>
    <origin xyz="0 0 0" rpy="1.5708 0 0"/>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1"/>
    </material>
  </visual>
</link>

<joint name="fl_wheel_joint" type="continuous">
  <parent link="base_link"/>
  <child link="fl_wheel"/>
  <origin xyz="0.1375 0.1575 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
</joint>

<!-- Front Right Wheel -->
<link name="fr_wheel">
  <visual>
    <geometry>
      <cylinder radius="0.048" length="0.04"/>
    </geometry>
    <origin xyz="0 0 0" rpy="1.5708 0 0"/>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1"/>
    </material>
  </visual>
</link>

<joint name="fr_wheel_joint" type="continuous">
  <parent link="base_link"/>
  <child link="fr_wheel"/>
  <origin xyz="0.1375 -0.1575 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
</joint>

<!-- Rear Left Wheel -->
<link name="rl_wheel">
  <visual>
    <geometry>
      <cylinder radius="0.048" length="0.04"/>
    </geometry>
    <origin xyz="0 0 0" rpy="1.5708 0 0"/>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1"/>
    </material>
  </visual>
</link>

<joint name="rl_wheel_joint" type="continuous">
  <parent link="base_link"/>
  <child link="rl_wheel"/>
  <origin xyz="-0.1375 0.1575 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
</joint>

<!-- Rear Right Wheel -->
<link name="rr_wheel">
  <visual>
    <geometry>
      <cylinder radius="0.048" length="0.04"/>
    </geometry>
    <origin xyz="0 0 0" rpy="1.5708 0 0"/>
    <material name="black">
      <color rgba="0.1 0.1 0.1 1"/>
    </material>
  </visual>
</link>

<joint name="rr_wheel_joint" type="continuous">
  <parent link="base_link"/>
  <child link="rr_wheel"/>
  <origin xyz="-0.1375 -0.1575 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  </joint>

</robot>
```

File: launch/display.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
import os

def generate_launch_description():
    pkg_path = get_package_share_directory('mecanum_description')
    urdf_file = os.path.join(pkg_path, 'urdf', 'mecanum_robot.urdf')

    with open(urdf_file, 'r') as f:
        robot_description = f.read()

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            parameters=[{'robot_description': robot_description}]
        ),
        Node(
            package='rviz2',
            executable='rviz2',
            arguments=['-d', os.path.join(pkg_path, 'rviz', 'config.rviz')]
        ),
    ])
```

File: rviz/config.rviz

```
Panels:
  - Class: rviz_common/Displays
    Name: Displays
Visualization Manager:
  Class: ""
  Displays:
    - Class: rviz_default_plugins/Grid
      Enabled: true
      Name: Grid
    - Class: rviz_default_plugins/RobotModel
      Enabled: true
      Name: RobotModel
      Description Topic:
        Value: /robot_description
    - Class: rviz_default_plugins/Odometry
      Enabled: true
      Name: Odometry
      Topic:
        Value: /odom
      Keep: 100
  Global Options:
    Fixed Frame: odom
    Frame Rate: 30
```

File: setup.py

```
from setuptools import setup
import os
from glob import glob

package_name = 'mecanum_description'

setup(
    name=package_name,
    version='0.0.1',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages', ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name, 'urdf'), glob('urdf/*')),
        (os.path.join('share', package_name, 'launch'), glob('launch/*')),
        (os.path.join('share', package_name, 'rviz'), glob('rviz/*')),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='Ricardo',
    maintainer_email='your@email.com',
    description='Mecanum Robot Description',
    license='MIT',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [],
    },
)
```

File: odom_to_tf.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from nav_msgs.msg import Odometry
from tf2_ros import TransformBroadcaster
from geometry_msgs.msg import TransformStamped

class OdomToTF(Node):
    def __init__(self):
        super().__init__('odom_to_tf')
        self.br = TransformBroadcaster(self)
        self.sub = self.create_subscription(Odometry, '/odom', self.odom_callback, 10)

    def odom_callback(self, msg):
        t = TransformStamped()
        t.header.stamp = self.get_clock().now().to_msg()
        t.header.frame_id = 'odom'
        t.child_frame_id = 'base_link'

        t.transform.translation.x = msg.pose.pose.position.x
        t.transform.translation.y = msg.pose.pose.position.y
        t.transform.translation.z = msg.pose.pose.position.z

        t.transform.rotation = msg.pose.pose.orientation

        self.br.sendTransform(t)

def main():
    rclpy.init()
    node = OdomToTF()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Build Instructions

```
cd ~/mecanum_viz_ws
colcon build
source install/setup.bash
```

Execution Steps

Terminal 1 - Hotspot:

```
nmcli connection up Hotspot
```

Terminal 2 - micro-ROS Agent:

```
docker run -it --rm --net=host microros/micro-ros-agent:humble udp4 --port 8888 -v4
```

Terminal 3 - Teleop:

```
cd ~/Desktop
source /opt/ros/humble/setup.bash
python3 teleop_qweasdzxc.py
```

Terminal 4 - RViz:

```
cd ~/mecanum_viz_ws
source install/setup.bash
QT_QPA_PLATFORM=xcb ros2 launch mecanum_description display.launch.py
```

Terminal 5 - Odom to TF:

```
source /opt/ros/humble/setup.bash
python3 ~/mecanum_viz_ws/src/mecanum_description/odom_to_tf.py
```

Portenta:

Press RESET button after agent is running.

RViz Visualization Tips

```
To enhance visualization in RViz:

1. Zoom: Use mouse scroll wheel
2. Add TF display: Shows coordinate frames
3. Add Path display: Shows robot trajectory (Topic: /odom)
4. Change background: Global Options -> Background Color -> Black
5. Change grid color: Grid -> Color -> Dark gray or green
6. Odometry arrows: Odometry -> Keep: 200 for more history

Save configuration: File -> Save Config As
```

Notes

- Robot orientation depends on IMU initial position at startup
- Reset Portenta with robot facing desired 'forward' direction
- Lateral movement may have slight drift (normal for mecanum)
- QT_QPA_PLATFORM=xcb fixes Wayland/RViz crash issue

Next Step

Step 5: Replace simple model with detailed 3D STL model of the actual robot, including animated mecanum wheels.