EE478
Introduction to Multi-disciplinary Robotics

Simulating Robot in Virtual Environment

(TA) Sungjae Min
School of Electrical Engineering
USRG, KAIST

March 27th, 2025

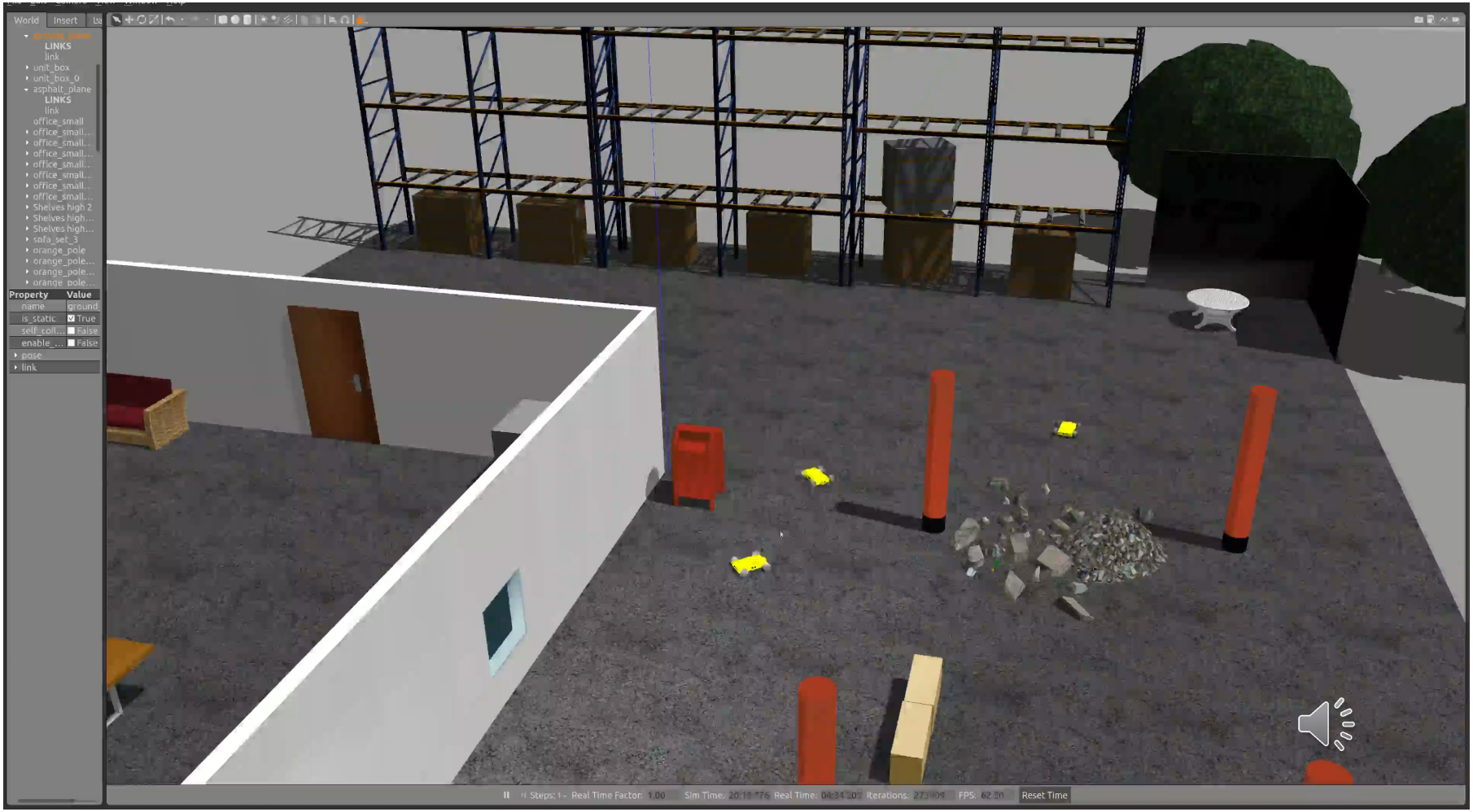
Simulating Robot in Virtual Environment

- Why simulate robot in virtual environment?
  - **Safety:** Testing in real environments can pose risks to the robot or surroundings, whereas simulation allows for risk-free experimentation across various scenarios.
  - **Cost Efficiency:** Building and modifying physical robots can be expensive, but simulations allow for easy modifications and testing in multiple environments without additional costs.
  - **Repeatability:** Enables repeated testing under identical conditions to validate the stability and efficiency of algorithms.
  - **Scalability:** Virtual environments or conditions that do not exist in the real world can be created to test robots, offering expansive testing capabilities.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

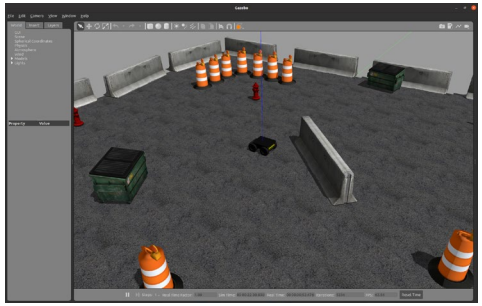
❖ Why simulate robot in virtual environment?



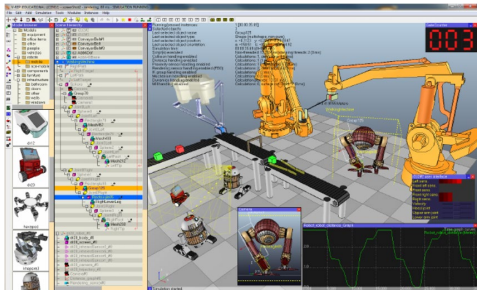
© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

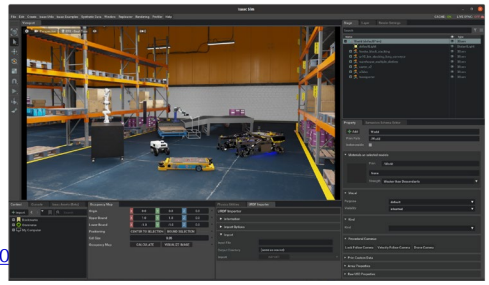
- Simulators commonly used with ROS?
  - **Gazebo:** Widely used in the robotics community, benefiting from large user base and extensive documentation.



  - **V-REP (CoppeliaSim):** Versatile simulation not just robots but also sensors, mechanism, and even human-robot interactions with various scripting language.



  - **Isaac Sim:** Ultra realistic simulation provides advanced tools for training and testing AI models, including AI for perception tasks.



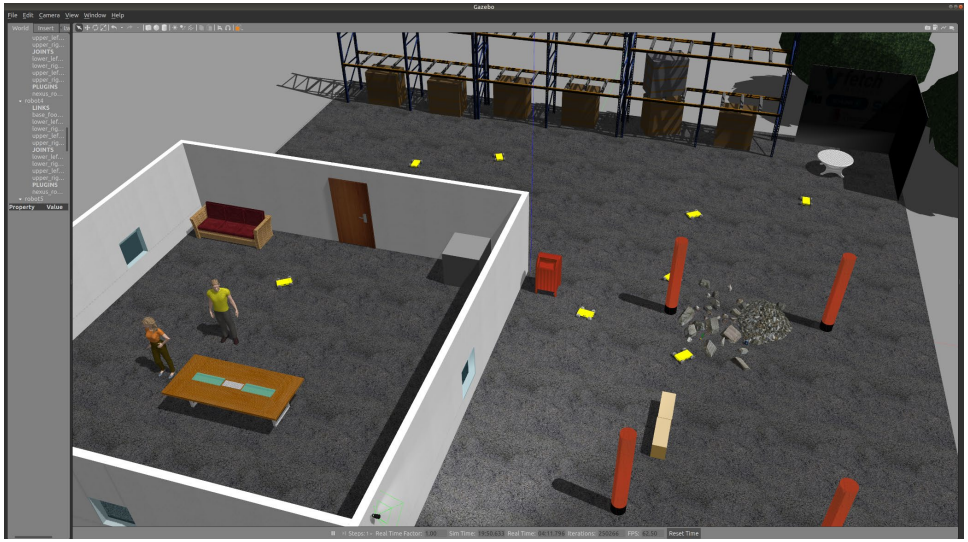
Gazebo example image: https://clearpathrobotics.com/assets/guides/foxy/husky/HuskySimulation.html
V-REP example image: https://www.researchgate.net/figure/Screen-shot-of-V-REPs-application-main-window-robot-models-are-courtesy-of-Lyall_fig6_220
Isaac Sim example image: https://docs.omniverse.nvidia.com/isaacsim/latest/index.html

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ Gazebo: Default simulator for ROS

- ❑ We will use Gazebo as a simulator for the mid-term simulation phase to test our AI-based autonomous rover.



- ❑ So we briefly introduce basic usage of Gazebo in this lecture
  - ➢ "World" in Gazebo
  - ➢ Obtain sensor data from the simulator
  - ➢ Control robot in the simulator through ROS

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ What is a "World" in Gazebo?

- ❑ Represents the simulation environment where your robots and object exist. It contains all information about your environment from physical properties, like gravity and lighting, and robots, sensors, and objects.

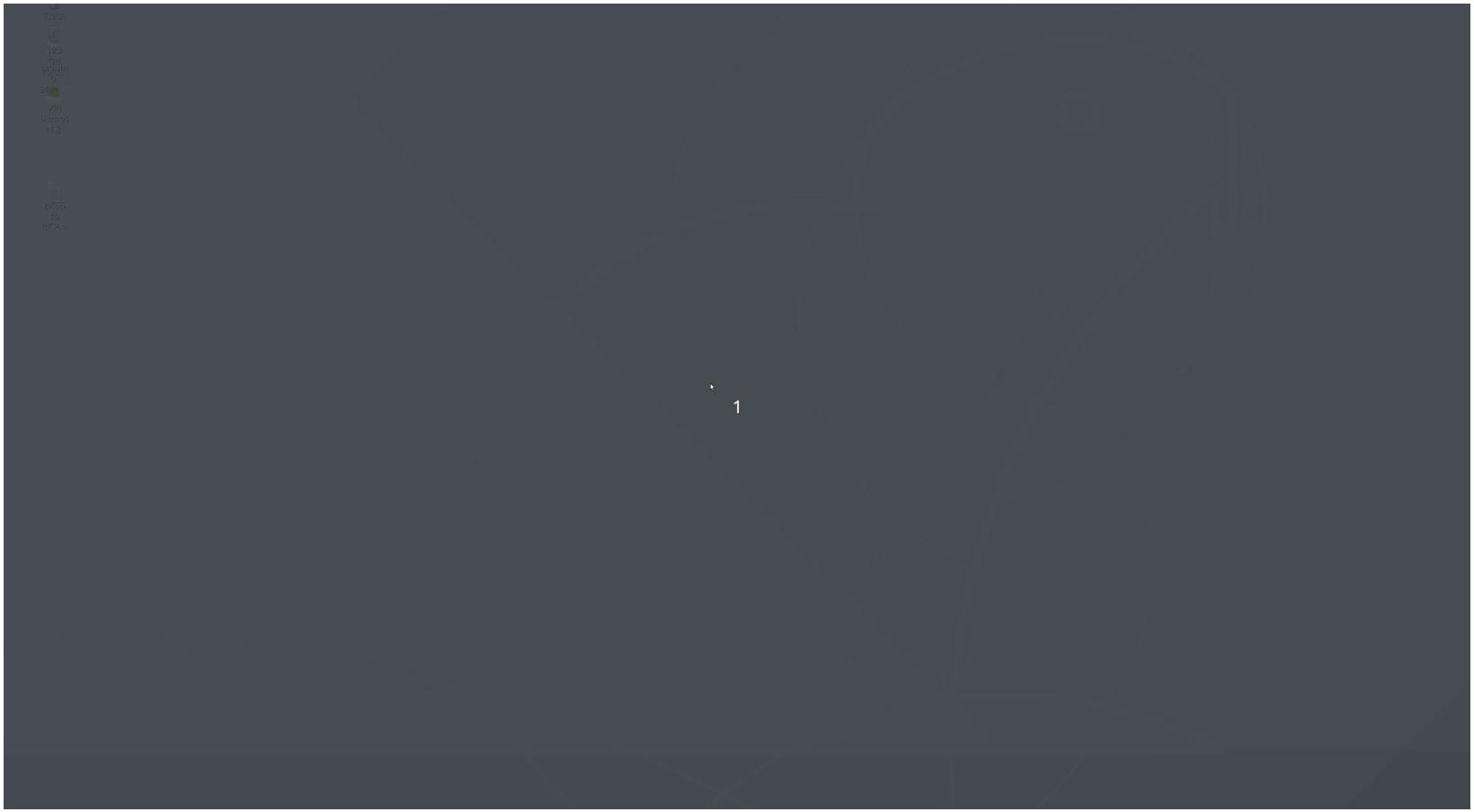
- ❑ Designing a "World" in simulation is crucial for realistic robot simulation. And it allows developers to test robots in environment and deploy to real-world ensuring safety, efficiency.

- ❑ Gazebo offers a wide range of tools for customizing your own "World" to fit specific needs.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **What is a "World" in Gazebo?**
  - How to make custom world?

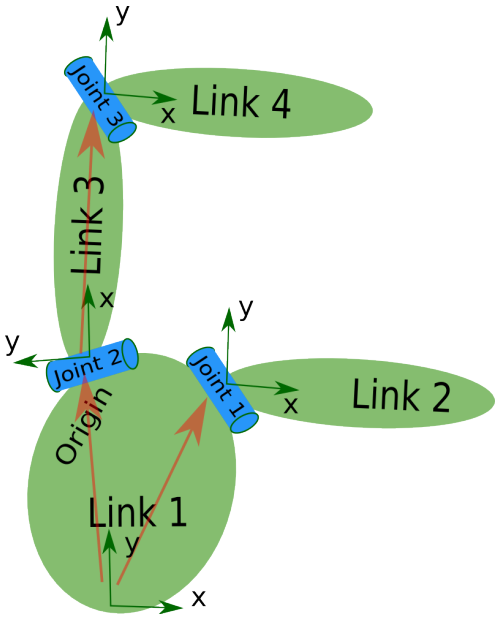


© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ How can we define robot model for simulation?

- ❑ URDF (Unified Robot Description Format)
  - ➢ XML file that can describes robot with single-tree kinematics.



Robot kinematics + Dynamics

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

  <!-- Base Link -->
  <link name="base_link">
    <visual>
      <geometry>
        <box size="0.6 0.4 0.2"/>
      </geometry>
    </visual>
  </link>
  …
  <!-- Arm Links -->
  <link name="Link_1"/>
  <link name="Link_2"/> …

  <!-- Joints for Arm -->
  <joint name="arm_base_joint" type="fixed">
    <parent link="base_link"/>
    <child link="arm_base"/>
    <origin xyz="0 0 0.2" rpy="0 0 0"/>
  </joint>

  <joint name="arm_joint_1" type="revolute">
    <parent link="arm_base"/>
    <child link="Link 2"/>
    <origin xyz="0 0 0.1" rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
    <limit lower="-1.57" upper="1.57" effort="10" velocity="1"/>
  </joint>
…
</robot>
```

robot.urdf

- ➢ We can describe our robot's simulation model with 3D mesh (such as stl or primitive shapes) and URDF!
  - ▪ https://wiki.ros.org/urdf/Tutorials

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **How can we define robot's model for simulation?**
  - xacro: XML macro developed for simplifying the scripting URDF.
    - **Reduce complexity**, **Parameterization**, **Modularity**

platform.urdf.xacro

realsense.urdf.xacro

robot.urdf.xacro

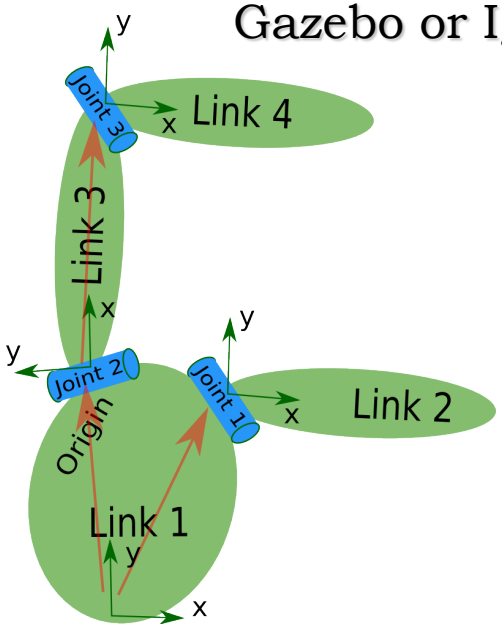
robot.urdf = xacro(robot.urdf.xacro)

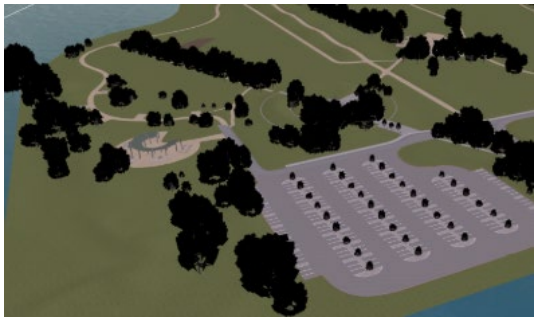
```
rosrun xacro xacro robot.urdf.xacro > robot.urdf
```


© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **How can we define world model for simulation?**
  - SDF (Simulation Description Format)
    - XML-based language commonly used by simulators such as Gazebo or Ignition to describe an entire simulation world.





```xml
<?xml version='1.0'?>
<sdf version="1.4">
  <model name="my_robot">
    <pose>0 0 0.5 0 0 0</pose>
    <static>true</static>
    <link name="link">
      <inertial>
        <mass>1.0</mass>
        <inertia>
          <ixx>0.083</ixx>
          <ixy>0.0</ixy>
          <ixz>0.0</ixz>
          <iyy>0.083</iyy>
          <iyz>0.0</iyz>
          <izz>0.083</izz>
        </inertia>
      </inertial>
      <collision name="collision">
        <geometry>
          <box>
            <size>1 1 1</size>
          </box>
        </geometry>
      </collision>
      <visual name="visual">
        <geometry>
          <box>
            <size>1 1 1</size>
          </box>
        </geometry>
      </visual>
    </link>
  </model>
</sdf>
```

Robot kinematics + Physics
+ Sensor configuration + World + Lighting

robot.sdf or world.sdf

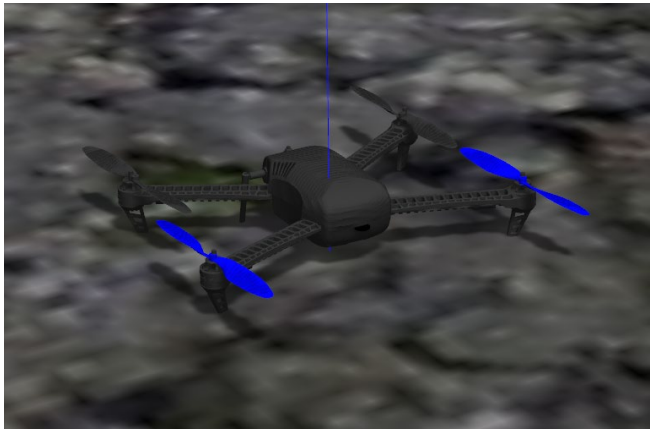
- SDF is essential for full-featured Gazebo simulations (robot + environment + sensors).
  - https://wiki.ros.org/urdf/Tutorials

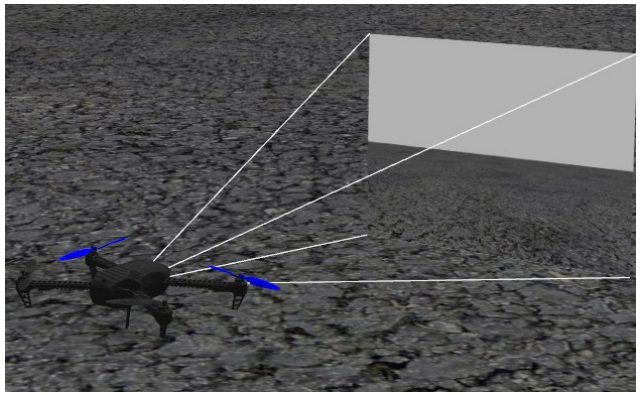
© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ **How can we define robot's model for simulation?**

- ❑ Using Multiple SDF Files for a Robot Platform and Camera.
  - ➢ **Reduce complexity**, **Parameterization**, **Modularity**





platform.sdf | camera.sdf | robot.sdf

```
<sdf version="1.7">
  <model name="robot">
    <include>
      <uri>file://platform.sdf</uri>
    </include>
    <include>
      <uri>file://camera.sdf</uri>
    </include>
  </model>
</sdf>
```


© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ Delve into simulation package

❑ Starting from the simulation launch file

➢ posix_sitl.launch

posix_sitl.launch M × iris_fpv_cam.sdf

launch > posix_sitl.launch > launch > arg

```xml
<?xml version="1.0"?>
<launch>
    <!-- Posix SITL environment launch script -->
    <!-- launches PX4 SITL, Gazebo environment, and spawns vehicle -->
    <!-- vehicle pose -->
    <arg name="x" default="0"/>
    <arg name="y" default="0"/>
    <arg name="z" default="0"/>
    <arg name="R" default="0"/>
    <arg name="P" default="0"/>
    <arg name="Y" default="0"/>
    <!-- vehicle model and world -->
    <arg name="est" default="ekf2"/>
    <arg name="vehicle" default="iris"/>
    <arg name="world" default="$(find mavlink_sitl_gazebo)/worlds/empty.world"/>
    <!-- <arg name="sdf" default="$(find mavlink_sitl_gazebo)/models/$(arg vehicle)/$(arg vehicle).sdf"/> -->
    <arg name="sdf" default="$(find mavlink_sitl_gazebo)/models/iris_fpv_cam/iris_fpv_cam.sdf"/>
```

- Launch File Overview
  Identifies the robot SDF and world file being used.
  Sets the initial spawn position and other parameters.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❖ Delve into simulation package

❑ Starting from the simulation launch file

➢ iris_fpv_cam.sdf

- Simple Structure
  The file mostly includes the iris platform SDF and the FPV camera SDF.
- Joint Definition
  It defines how the camera is attached to the platform using a joint.
- Detailed SDF Files
  The actual platform and camera files have more complex implementations.
- Students can explore them for deeper understanding.

sitl.launch M iris_fpv_cam.sdf ×

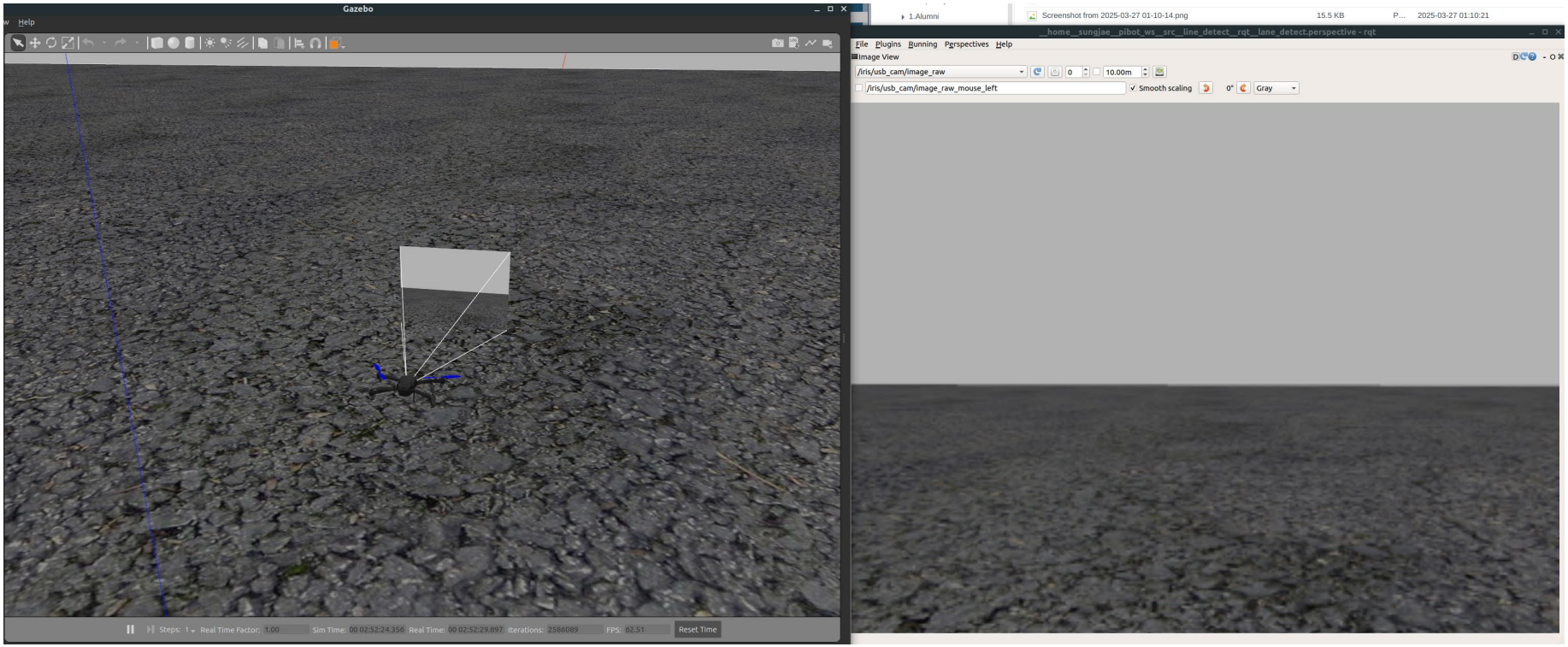
itl_gazebo > models > iris_fpv_cam > iris_fpv_cam.sdf > sdf > mo

```
<?xml version='1.0'?>
<sdf version='1.5'>
  <model name='iris_fpv_cam'>

    <include>
      <uri>model://iris</uri>
    </include>

    <include>
      <uri>model://fpv_cam</uri>
      <pose>0 0 0 0 0 0</pose>
    </include>
    <joint name="fpv_cam_joint" type="fixed">
      <child>fpv_cam::link</child>
      <parent>iris::base_link</parent>
      <axis>
        <xyz>0 0 1</xyz>
        <limit>
          <upper>0</upper>
          <lower>0</lower>
        </limit>
      </axis>
    </joint>

  </model>
</sdf>
```


© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

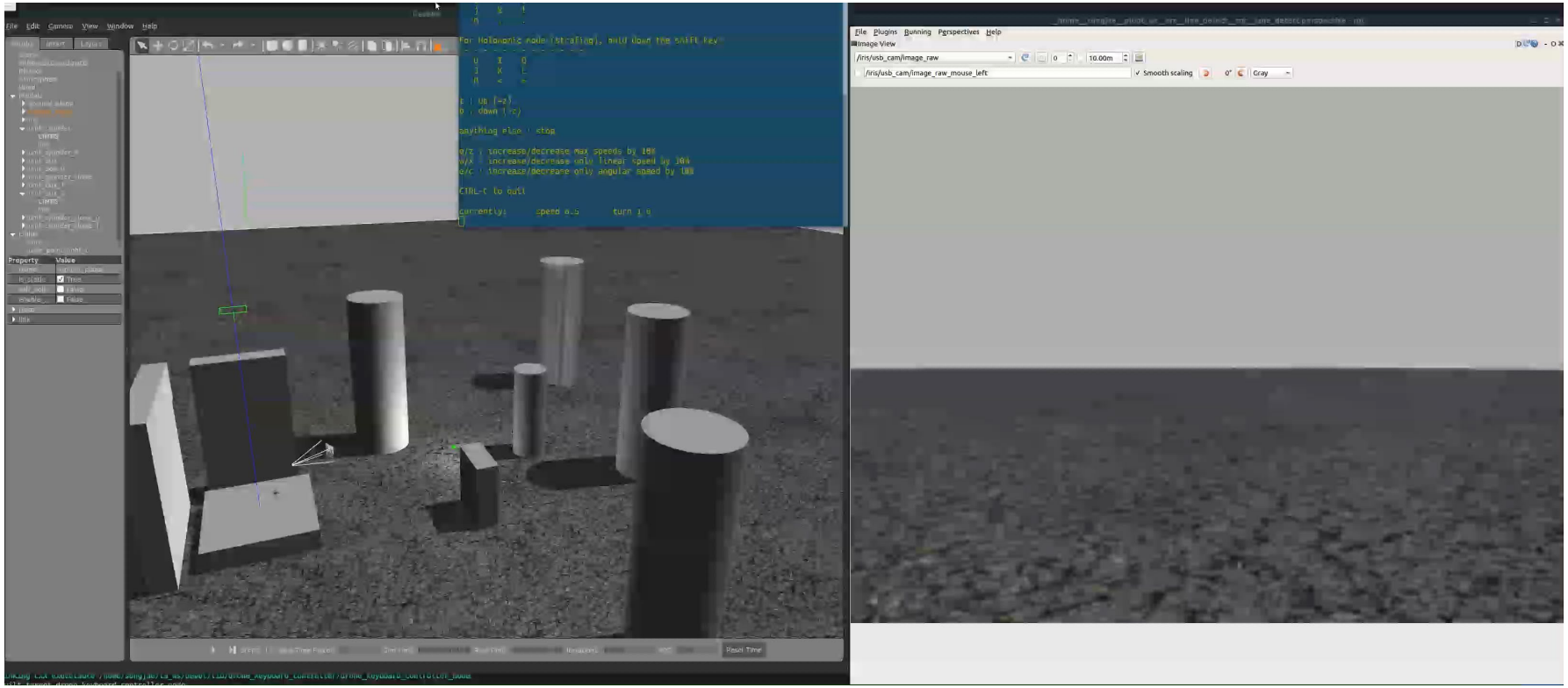
- **Delve into simulation package**
  - **Starting from the simulation launch file**
    - Run modified launch file



© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**



© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

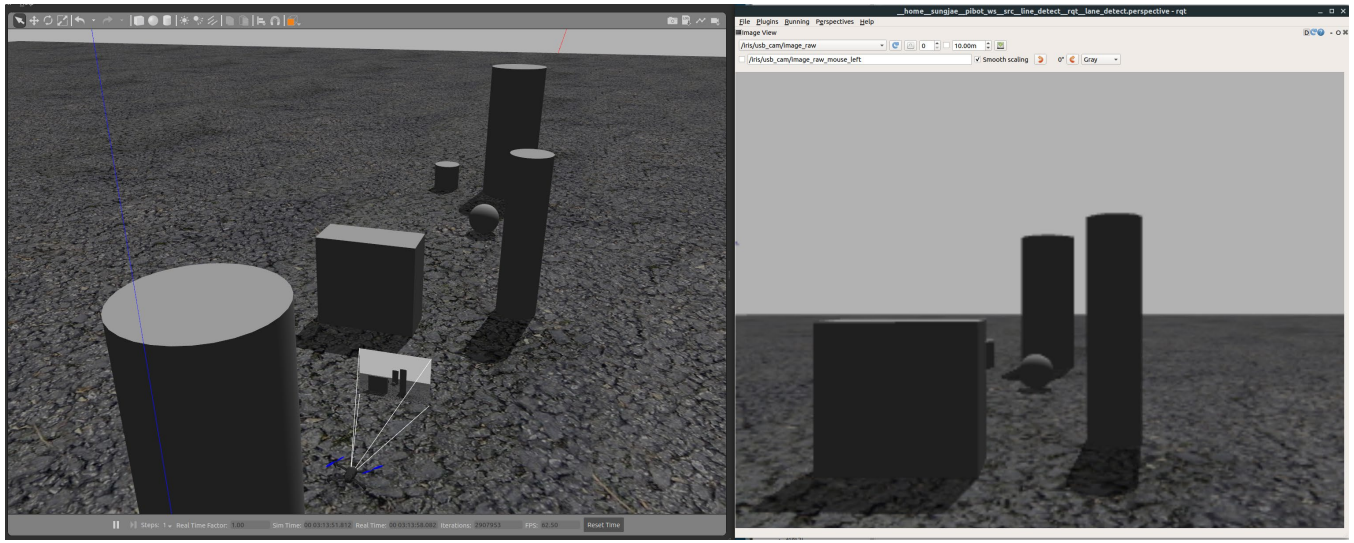
- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 1: Create Own Simplified Virtual Environment that mirrors a real-world room you're familiar with.** (15 pts)
    - **1)** Select an environment (e.g., room, hallway, maze) and either take a photograph or find an image online.
    - **2)** Construct a simplified virtual world in Gazebo that resembles the chosen image.
    - **3)** Save your custom environment file and the reference image in package/worlds.

    - You may utilize object libraries available on the internet to construct your environment, such as:
      https://github.com/osrf/gazebo_models
      https://github.com/leonhartyao/gazebo_models_worlds_collection
    - The environment doesn't need to be precisely detailed or densely populated with objects. Aim to use about 10 to 20 objects.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 2: Spawn the Drone with an FPV Camera and Simulate Your Robot in Gazebo**. (5 pts)
    - **1)** Spawn the Drone in the world you created for Problem 1.
    - **2)** Verify the Robot's Appearance and check the camera feed using rqt_image_view.
    - **3)** Check the Drone's Spawn Position to ensure it does not collide with any objects.



© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 3: Develop a Robot Command Interface Package** (20pts)
    - **1)** Use catkin_create_pkg to start developing a new ROS package.
    - **2)** Create a node that subscribes to an input command topic and publishes velocity commands to "/mavros/setpoint_velocity/cmd_vel".
    - **3)** Run Gazebo, MAVROS, and your newly developed package together to simulate your robot.

    - Tip: Use teleop_twist_keyboard package to make input command from keyboard and it will publish "/cmd_vel"

```
sudo apt install ros-noetic-teleop-twist-keyboard
```


© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 4: Teleoperate the Drone Solely Through Sensor Feedback.** (20pts)
    - **1)** Open rqt_image_view and teleop_twist_keyboard, and make it full screen so that the Gazebo screen is not directly visible.
    - **2)** Manually control your robot using the keyboard teleoperation node to avoid collisions with objects. Capture video footage of the screen while completing a round trip in your environment.
    - **3)** Open the "rqt_graph" and check your system diagram.

rqt_graph

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 5: Expanding Sensor Usage and Visualization for teleoperation.** (30pts)
    - **1)** Choose a Drone Model with Additional Sensors
      - The provided PX4 simulation package includes various drone models and sensor configurations (e.g., RGB camera, depth camera, LiDAR).
      - Select a model that best suits your environment and exploration needs.
    - **2)** Set Up RViz for Visualization
      - To visualize more than just camera feeds (e.g., LiDAR or other sensor data), you need a 3D visualization tool like Rviz.
      - Configure RViz so you can teleoperate the drone with full sensor feedback.
    - **3)** Fly Your New Drone Using RViz (or RQT)
      - Rely on RViz or RQT for sensor views and control.
      - Fly the drone conveniently through your custom environment and complete a full lap without collisions.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - **Problem 6: Visualize the Drone's Flight Path.** (10pts)
    - **1)** Why Visualize the Flight Path?
      - Understanding the current position is crucial in autonomous systems.
      - Tracking the drone's path helps you see where it has been and plan where it needs to go next.
      - Display the path in Rviz so you can reference it while teleoperating the drone.
    - **2)** Implementation Options
      - You may add path visualization to the node you developed in Problem 3, or
      - Create a new package and a separate node dedicated to trajectory visualization.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

- **HW: Building own environment and customize robot and teleoperation using feedback camera image**
  - Please submit your work on the robot simulation via KLMS [DUE: 2025.04.13 23:59]
  - Submission Requirements:
    - The world file you have created.
    - The modified launch file.
    - Developed controller interface package.
    - Video files capturing the teleoperation sessions.
    - A short report (1-2 pages) that includes:
      - A summary of what you have learned.
      - The modifications you made to the launch and sdf file.
      - Describe your system with the system graph shown in "rqt_graph".
      - References to the object library for Gazebo that you utilized.

Make a single .zip file of your implemented code package folder!

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❑ Creating new package (catkin_create_pkg)

- Command structure

```
catkin_create_pkg <package_name> [dependencies...]
```

  - This creates a directory named my_robot_pkg containing:
    package.xml
    CMakeLists.txt
    and any other files needed for a basic ROS package setup.

```
sungjae@USRG-MSJ-DESKTOP:~/ta_ws/src$ catkin_create_pkg drone_keyboard_controller roscpp std_msgs geometry_msgs
Created file drone_keyboard_controller/package.xml
Created file drone_keyboard_controller/CMakeLists.txt
Created folder drone_keyboard_controller/include/drone_keyboard_controller
Created folder drone_keyboard_controller/src
Successfully created files in /home/sungjae/ta_ws/src/drone_keyboard_controller. Please adjust the values in package.xml.
sungjae@USRG-MSJ-DESKTOP:~/ta_ws/src$ ls
apriltag-imgs  CMakeLists.txt  drone_keyboard_controller  true_rt_tag
```

```
sungjae@USRG-MSJ-DESKTOP:~/ta_ws/src/drone_keyboard_controller$ ls
CMakeLists.txt  include  package.xml  src
```

  - Dependencies
    Specify which packages you depend on (e.g., roscpp, rospy, geometry_msgs).
    The script automatically updates package.xml and CMakeLists.txt.

https://wiki.ros.org/ROS/Tutorials/CreatingPackage

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

❑ Creating new package (catkin_create_pkg)

- Next Steps
  Navigate to the "drone_keyboard_controller" folder to start developing your nodes, launch files, etc.

  Remember to update both CMakeLists.txt and package.xml so your catkin workspace can build everything correctly.

```
cd ~/catkin_ws
catkin_make
source devel/setup.bash
```

- Folder Structure
  src/: Place your C++ or Python source files.
  scripts/ (optional): A common place for Python scripts if you prefer.
  launch/ (optional): Where launch files go if you have any.

© [2024] [Sungjae Min @ USRG, KAIST]

Simulating Robot in Virtual Environment

Thank you!

© [2024] [Sungjae Min @ USRG, KAIST]