Introduction

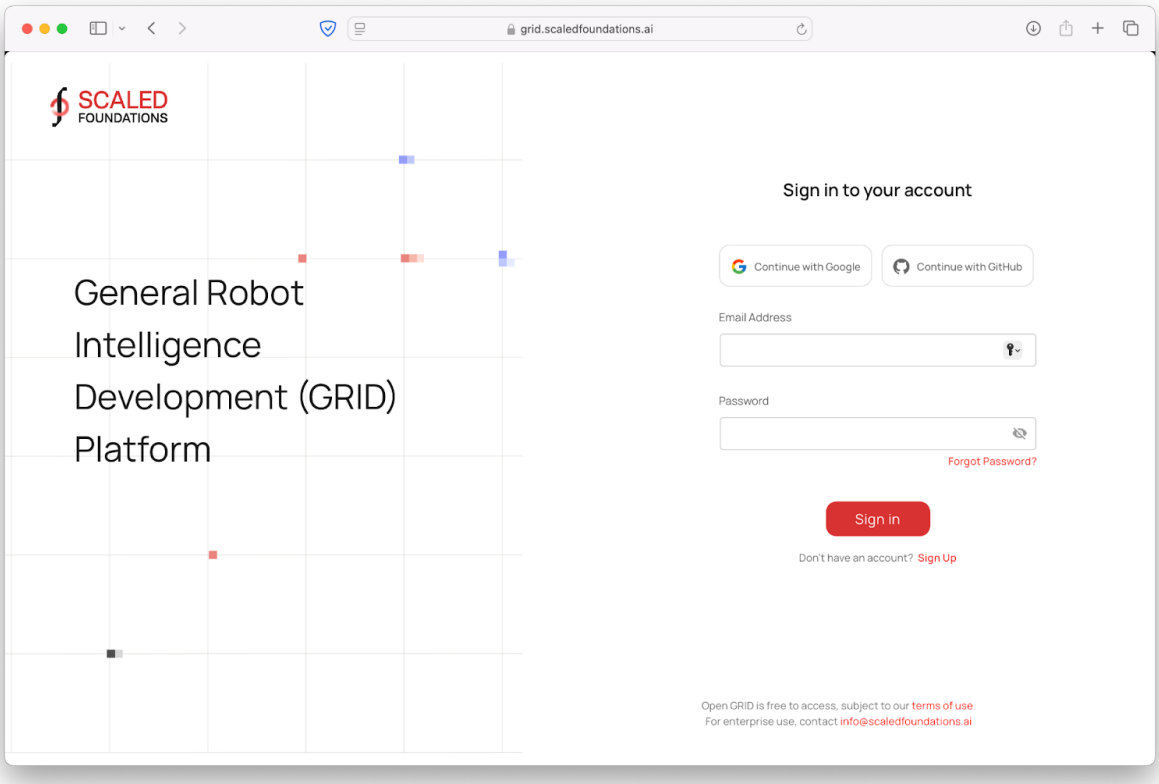
Welcome to the “Introduction to Robot Simulation and Intelligence” workshop! In this workshop, we will be using the GRID platform to simulate, develop, and deploy intelligent skills on a legged robot.

GRID is a comprehensive integrated development environment for robotics, consisting of high-fidelity simulation, AI foundation models, and development/deployment workflows. The version of GRID we will be using in the session, Open GRID, is a fully browser-based robot AI development platform.

GRID integrates NVIDIA Isaac Sim among other simulators, and in the session, we will be focusing on simulating and deploying intelligent skills for a legged robot (Unitree Go2) from Isaac Sim.

Get Started

Please navigate from your browser to grid.scaledfoundations.ai and sign up for an account. You can either use your Google/GitHub accounts for single sign-on, or feel free to set up an account using an email address and password.



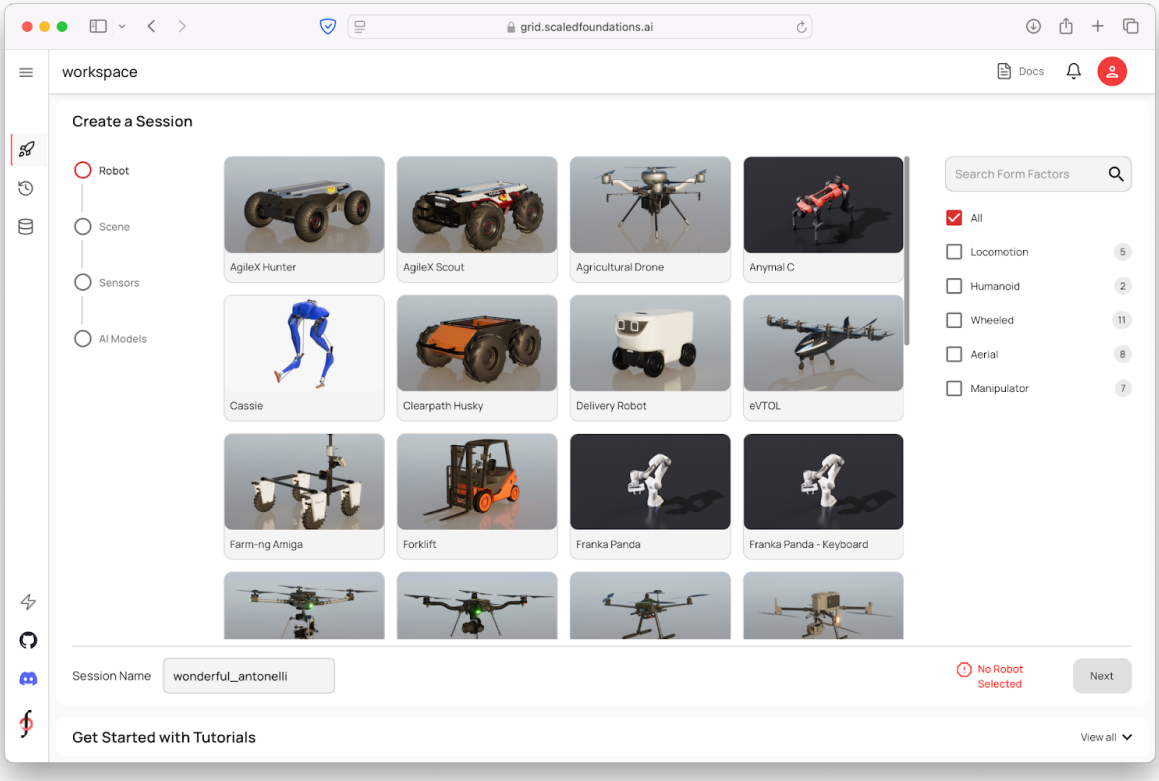
You will receive a verification email upon sign up, and once you verify your email through the link within, you will have access to the platform.

Understanding the GRID platform

Open GRID is a fully web-based platform to develop, train, validate and deploy intelligent skills for a variety of robots.

The Workspace is where you configure and launch your user sessions. It serves as a starting point for the development environment where you can select the following:

- Robot Configuration: Choose your robot
- Simulation Environment: Select and configure (if needed) your virtual environment.
- AI Models: Integrate AI models for various tasks and capabilities.



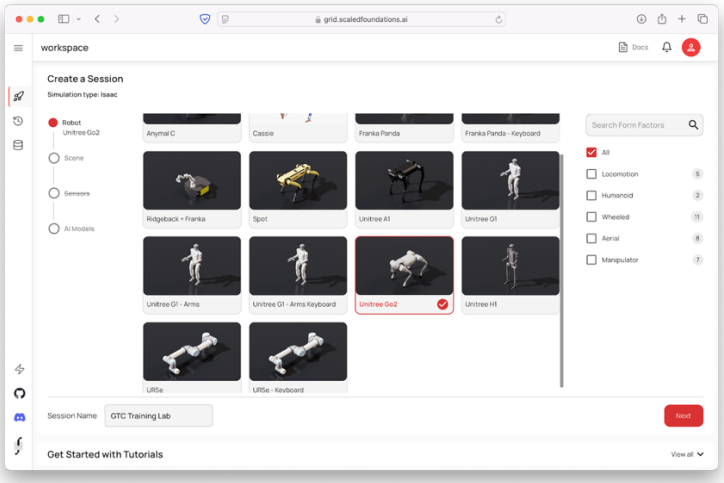
All robots start with a pre-configured set of sensors such as RGB, depth camera etc.

Configuring a Custom Session – Unitree Go2 in Warehouse

To create a custom session, follow these general steps:

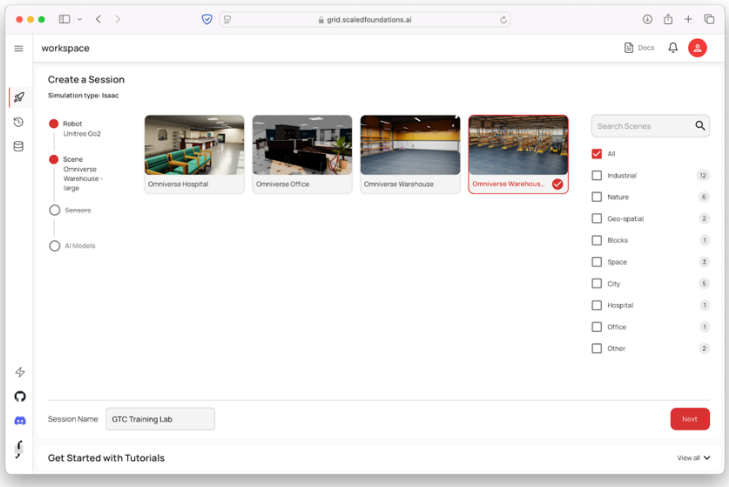
1. Select a robot

Select the Unitree Go2 quadruped/legged robot.



2. Select a Scene

GRID contains a variety of environments: urban settings through Google or Bing Maps, natural scenes like beaches or forests, or specialized locations like oil rigs, warehouses, and construction sites. The list of available scenes differs based on the chosen simulator/robot. For this session, choose the “Omniverse Warehouse - large” environment.



3. Choose AI Models

Optionally, preconfigure the AI models that will be used in your simulation. Navigate to the ‘AI Models’ tab for selection – you can choose from VLMs, segmentation, detection, RGB to depth, tracking, VO/SLAM models and more. If you have not decided on AI models yet, you can still import them after starting the session.

Developing within the GRID Session

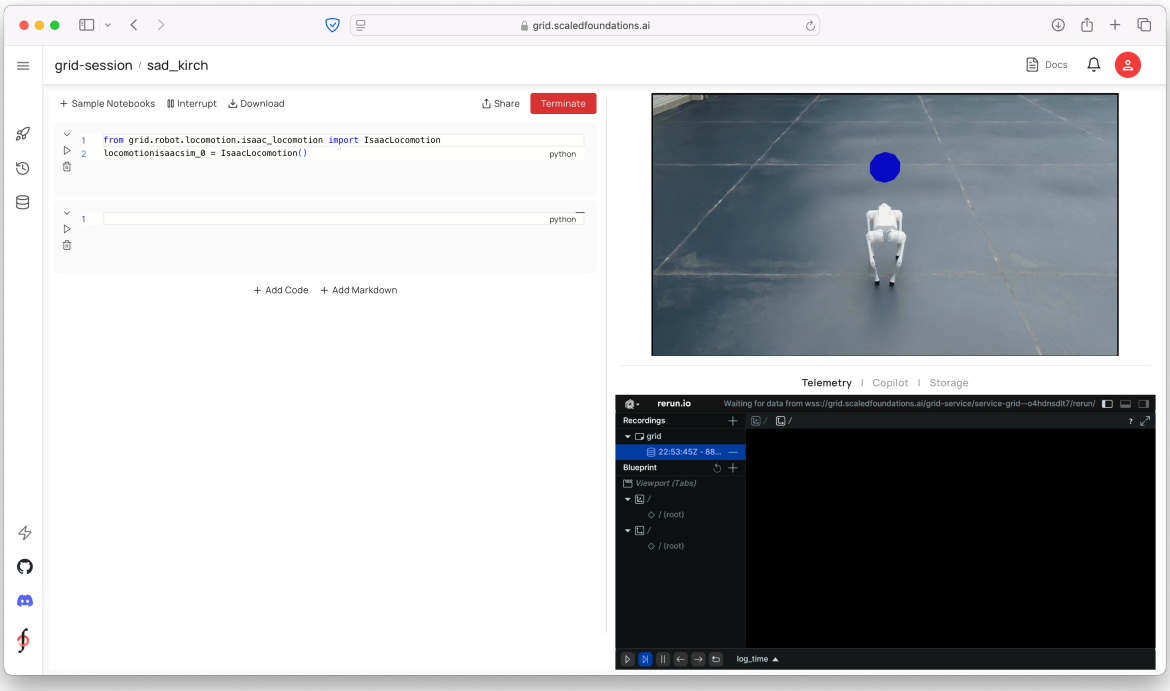
In this tutorial, you’ll learn how to use the GRID session to work with the robot and build skills. We’ll cover:

- **Session UI overview:** Understand the different elements in the session UI and how to use them.
- **Robot Initialization:** Setting up robot using GRID + Isaac Sim.
- **Sending Control commands:** Send velocity commands to move the robot within the scene.
- **Data Capture:** Retrieving and logging RGB and depth images along with other data.
- **Perception Models:** Running visual language, segmentation, depth, and object detection models.

---

0. Session UI Overview

Once launched, the GRID session has mainly three panels.



Python notebook (left): Essentially a Jupyter notebook, where you can interact with the GRID API for robot control, data capture, and AI models. You can also download any custom packages (!pip install) if needed.

Real-time simulation streaming (top right): See the Isaac Sim stream in real time. Please note that upon launch, it might take a while for the stream to come up depending on the complexity of the scene loaded in Isaac Sim/Omniverse.

Telemetry / Copilot / Storage: This panel has three tabs: the telemetry tab shows a Rerun window where you can log and visualize any data stream from the session – such as robot positions, 2D images from the sensors or AI model outputs, or 3D point clouds or maps etc. Copilot consists of an LLM integration to send text commands to the robot (which then results in code generation). Storage tab is a view of the cloud storage corresponding to the current session, where users can save and retrieve any files.

---

1. Robot Initialization

First, import the module corresponding to the robot and start the robot's control interface. This gives you access to methods for commanding movement and accessing camera data. For convenience, this cell will be preloaded upon session start.

```
from grid.robot.locomotion.isaac_locomotion import IsaacLocomotion

# Initialize the robot locomotion interface.
locomotionisaacsim_0 = IsaacLocomotion()
```

---

2. Sending Control commands

Next, define a certain velocity and have the robot move forward. Note that the commands latch – so until you send another velocity command or stop the robot, it will continue in the commanded direction.

```
from grid.utils.types import Velocity

# Define velocities.
linear_velocity = Velocity(1, 0, 0)
angular_velocity = Velocity(0, 0, 0)

# Command the robot to move forward.
locomotionisaacsim_0.moveByVelocity(linear_velocity, angular_velocity)
```

Use the angular velocity if you wish to rotate the robot.

```
angular_velocity = Velocity(0, 0, 1)
# Command the robot to turn counter-clockwise.
locomotionisaacsim_0.moveByVelocity(linear_velocity, angular_velocity)
```

Stop the robot using the ‘stop()’ function.

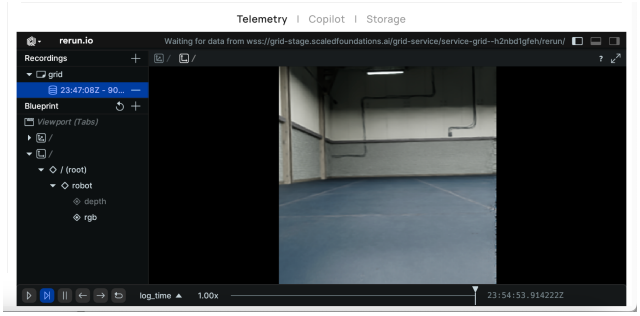
```
locomotionisaacsim_0.stop()
```

3. Accessing camera data and other ground truth

This section shows how to acquire images from both the RGB and depth cameras, which will be used as input for the perception models.

```
# Capture an RGB image.
rgb_image = locomotionisaacsim_0.getImage()

# Log the RGB image to Rerun
from grid.utils.logger import log
log("robot/rgb", rgb_image)

# Capture and log a depth image.
depth_image = locomotionisaacsim_0.getImage("camera_depth_0")
log("robot/depth", depth_image)
```

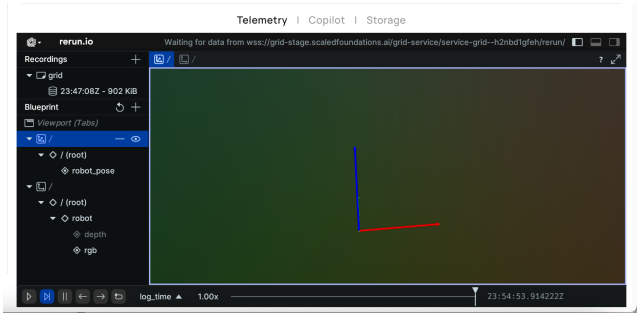
You should now be able to see the images come up in Rerun.



We can also obtain the world pose of the robot at any time.

```
from grid.utils.types import Position, Orientation, Pose

pose = Pose(locomotionisaacsim_0.getPosition(), locomotionisaacsim_0.getOrientation())
print(pose)
log("robot_pose", pose)
```

You can also view this in Rerun, but by opening the 3D view on the top bar (the icon on the left) 



4. Accessing AI Models

Leverage the GRID AI models to process the sensor data. Each model provides a different type of environmental understanding. In this section, we provide some examples of how to use models of different categories. Feel free to choose and play with other models or combine these models in creative ways to achieve intelligent capabilities!

4.1 Visual Language Model: MoonDream

Use the MoonDream VLM to interpret the scene by answering a natural language question based on the RGB image. GRID allows you to access state of the art vision-language intelligence in just 2-3 lines of code.

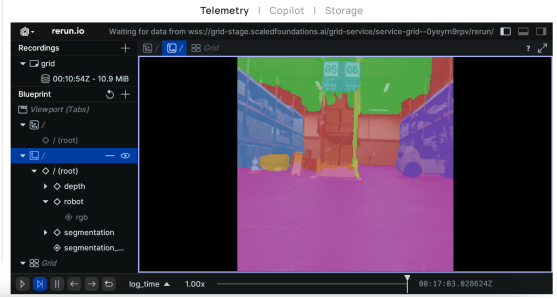
```
from grid.model.perception.vlm.moondream import MoonDream

vlm = MoonDream()
vlm.run(rgb_image.data, "What do you see?")
```

```
4   vlm.run(rgb_image.data, "What do you see?")
5                                                                    python

' The image features a warehouse with a yellow forklift parked in the middle of the
room. The forklift is positioned near the center of the warehouse, surrounded by
various items and equipment. \n\nIn addition to the forklift, there are multiple boxes
scattered throughout the warehouse, both on the floor and on the shelves. A fire
extinguisher can also be seen in the background, further emphasizing the industrial
nature of the space.'
```

3.2 Segmentation Model: OneFormer

Segment the scene to distinguish different objects or regions. This code block performs panoptic segmentation, which returns all the categories visible.

```
from grid.model.perception.segmentation.oneformer import OneFormer
seg_model = OneFormer()
seg_mask = seg_model.run(rgb_image.data, mode="panoptic")

import rerun as rr
rr.log("segmentation_model", rr.SegmentationImage(seg_mask))
```

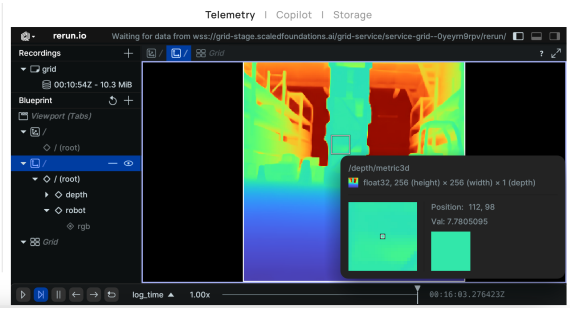


3.3 Depth Estimation Model: Metric3D

Use a neural network to generate a depth map from a monocular RGB image to understand the distance to various parts of the scene. (Tip: it would be interesting to compare this with the ground truth depth to see how well the model performs!)

```
from grid.model.perception.depth.metric3d import Metric3D
```

```
depth_model = Metric3D()
depth_image = depth_model.run(rgb_image.data)
rr.log("depth_model", rr.DepthImage(depth_image))
```



3.4 Object Detection Model: OWLv2

Detect specific objects—in this example, a forklift—from the RGB image.

```
from grid.model.perception.detection.owlv2 import OWLv2
det_model = OWLv2()

boxes, scores, labels = det_model.run(rgbimage=rgb_image.data, prompt="forklift")
```

The forklift might not be visible directly from where the robot is. Try to combine this with rotation or other movement commands to 'search' for the forklift!

Building Perception-Action skills

During the session, you will learn further how to use these fundamental capabilities to build more complex skills. For reference, you can also view the notebooks corresponding to these skills under the Sample Notebooks tab. For example, use the Detect and Navigate notebook to find and detect the forklift.