Development of an Autonomous Simulation, Testing and Data Generation Framework for Mobile Robots within Randomly Generated Plausible Scenarios

Master’s Thesis LEM-MA23/12

Satyam Uttamkumar Dudhagara

Mr. Benjamin Blumhofer

Agenda

Introduction and Motivation

Solution Needed: To address limitations of physical robot dataset generation and testing.

The need for robust **path planning** algorithms is paramount among navigation challenges with AI integration and thus the need for real AMR path training datasets.[3]

The need for quick and reliable testing of **navigation algorithms and sensor integration** in various environments has become significant.

The **reliance on AI** in mobile robotics is driving a growing demand for extensive and dependable **training datasets**.[2]

Robotics Testing Paradigm: Traditionally **physically demanding** or repetitive for **testing.**[1]

They are required for **versatile applications across industries** due to advances in **robotics, AI, and sensors**.[4]

Autonomous mobile robots (AMRs) have to **navigate various environments** (both **static** and **dynamic**).

Research Objectives

Resources: Utilize available **3D models** for **environment, assets, and robots.**

Generate **plausible factory/warehouse Scenarios** with provided resources.

Goal Stations: Provide world-referenced or model-referenced **goal poses**.

Generate **mobile robot taskset** with provided goal stations.

Capture **simulation time, robot pose, velocity, Navigation behavior tree status, contact/collision data, taskset status labels, and rosbags.**

Asses the dataset for collision and contacts made or unknown stoppages during the simulation taskset.

Create an autonomous **framework** utilizing the **NVIDIA Omniverse Isaac Sim** with the **digital twin concept** as its foundation to **generate mobile robot path datasets** and **navigation testing**.

Mobile Robot Navigation with Nav2 in ROS 2

- ROS2 Workspace
- NVIDIA Isaac Sim

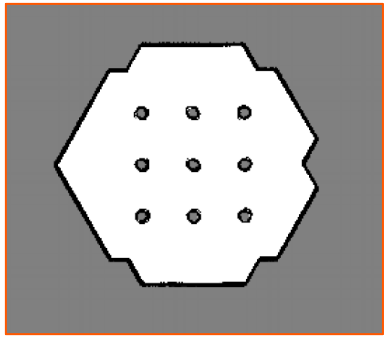


Figure 1. Example navigational map [5]

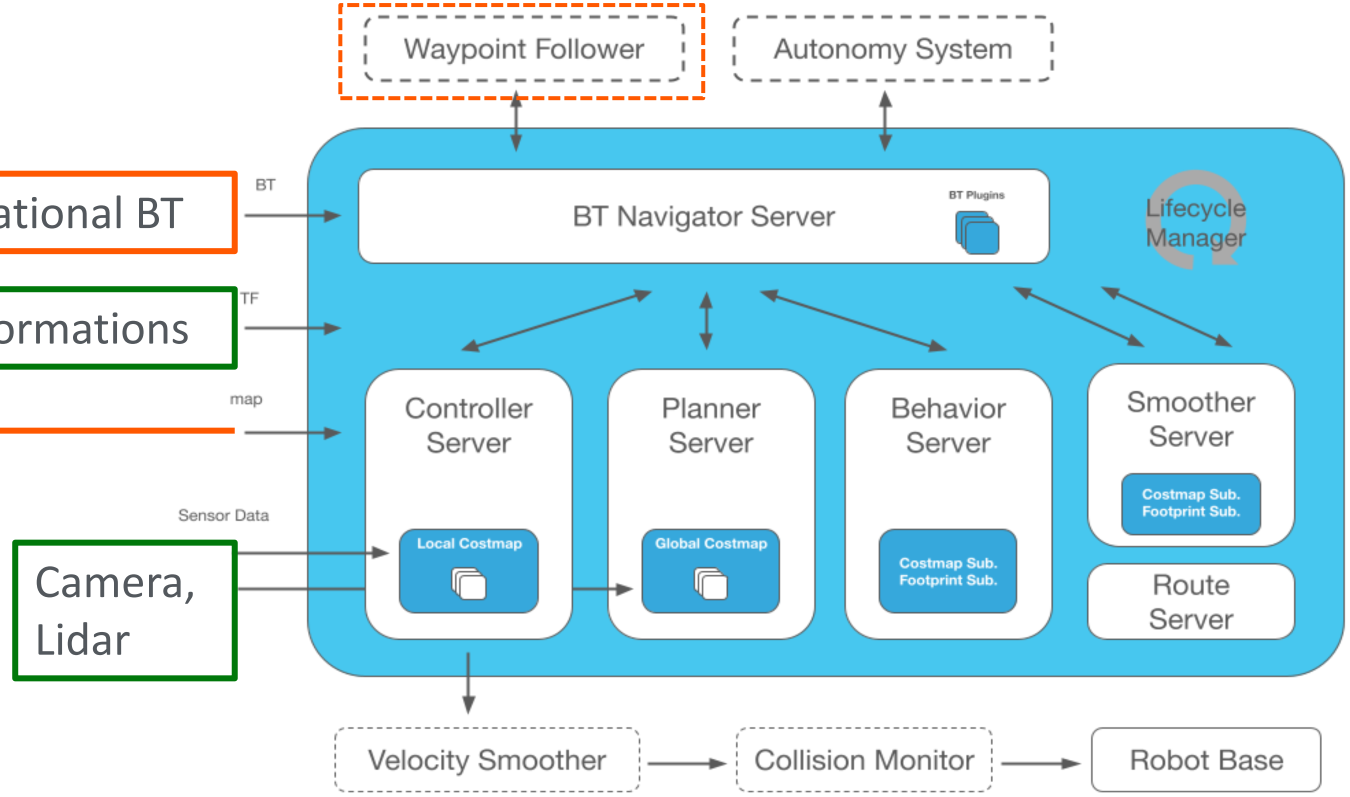


Figure 2. Architecture of Nav2 [6]

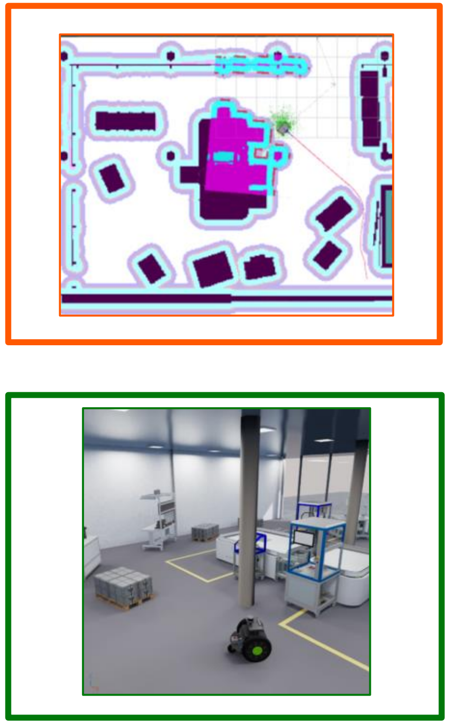


Figure 3. Navigation in Simulation: Rviz2 (Top); Isaac Sim (Bottom)

NVIDIA Isaac Sim as Simulator solution

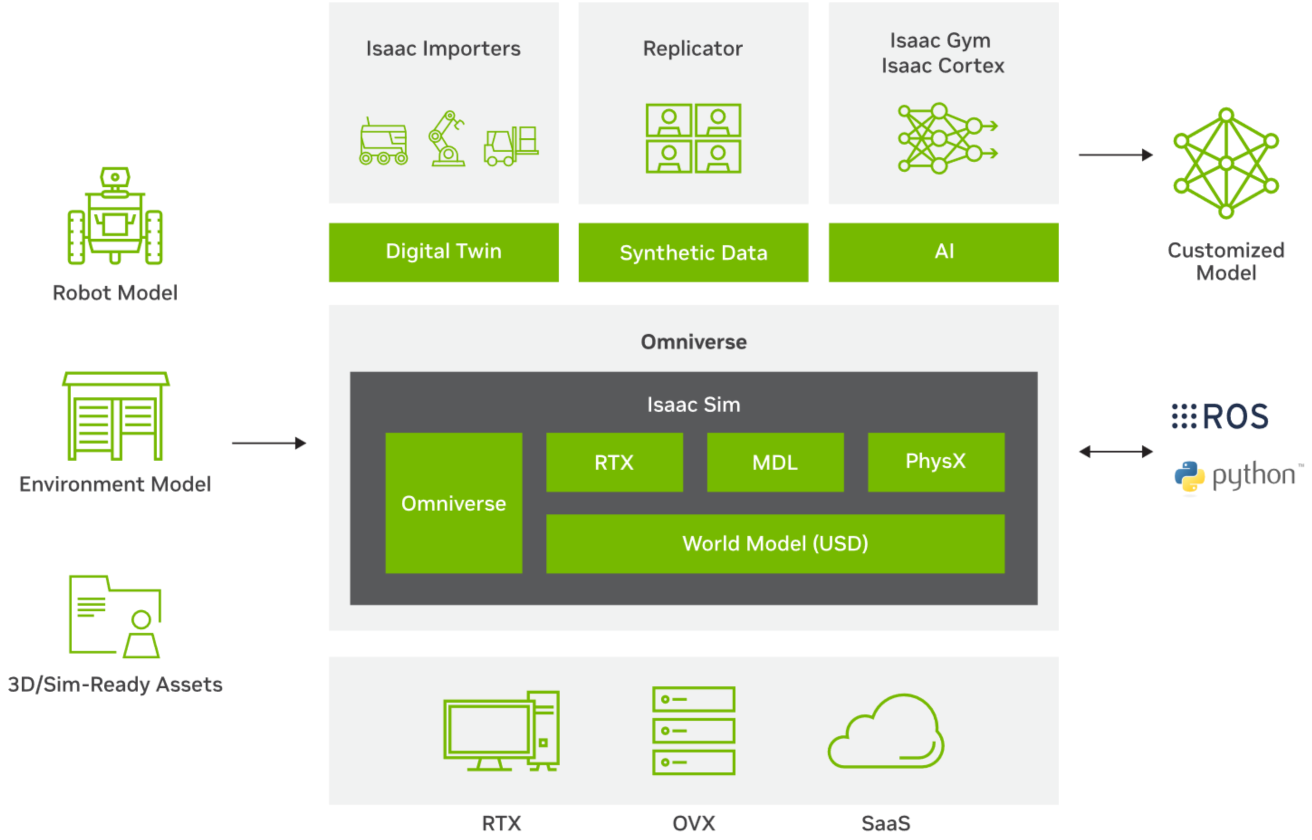


Figure 4. NVIDIA Isaac Sim Solution Stack [7]

Requirements and Environment

Framework Requirements: We aim to automate mobile robot testing in diverse simulation environments using NVIDIA Isaac Sim, the Isaac Sim API, and ROS 2, with a focus on recording synthetic data.

Framework Functionality: Provide autonomy and synchronization in simulation as well as ROS 2 navigation application

Variation Functionality: User-controlled plausible scene generation and robot taskset generation

Data Recording: Data significant for research/testing to needs to be recorded from these simulations.

Framework Development Environment:

NVIDIA Isaac Sim:
Python 3.7 Environment
ROS2 Bridge enabled

ROS2:
Python 3.8 dependent
ROS2 Foxy
Nav2 Installed

Ubuntu 20.04 LTS (64-bit Linux system)

RTX Workstation

Framework Design

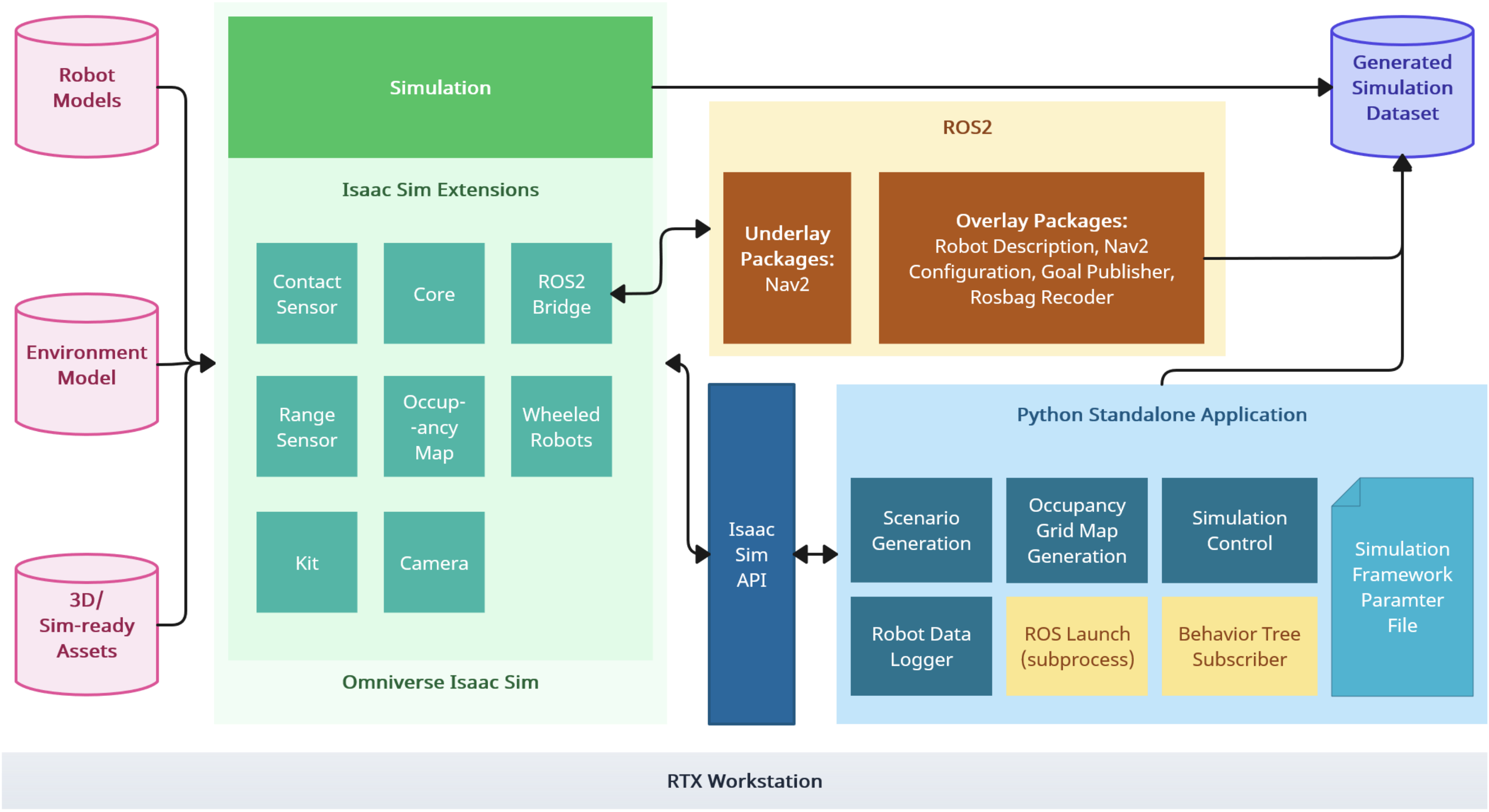


Figure 6. Architecture of the Framework

Framework Development

Simulation Setup → Plausible Scenario Generation → Map Generation → Goal Randomizer → ROS 2 Setup → ROS 2 Launch → Sim-ROS clock synchronization → Data Logger

- The phases of development of the framework using the Isaac Sim Extensions API.
- These phases in developing the framework utilize the resources from **NVIDIA's default asset library.**
- The framework is then implemented on the ***Smartfactory Lab*** Environment and 3D Sim-ready assets.

Werkzeugmaschinen und Steuerungen KAISERSLAUTERN

Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Framework Development

Simulation Setup → Plausible Scenario Generation → Map Generation → Goal Randomizer → ROS 2 Setup → ROS 2 Launch → Sim-ROS clock synchronization → Data Logger

An instance of SimulationApp (according to CONFIG)

A stage in Isaac Sim

Enable Physics

Set Physics Time-step

Set Stage unit scale

Create a world

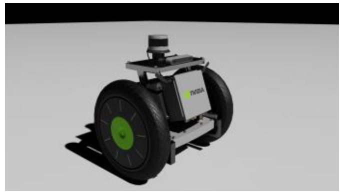
Define Callback functions

Error Handling and Cleanup

Framework Development

Simulation Setup → Plausible Scenario Generation → Map Generation → Goal Randomizer → ROS 2 Setup → ROS 2 Launch → Sim-ROS clock synchronization → Data Logger

1. Define **environment, asset and robot model** paths



2. Load the main **Environment**

3. **Define Domain of Randomization** of spawning objects

Positional boundaries and Orientation limits

4. **Generate Plausible scene** by populating environment with assets

Saving the Scene:

- **Reset** the simulation **world** and **update** the **simulation app**.
- Save the modified stage as a **USD file** for future use and analysis.

Framework Development

Simulation Setup → Plausible Scenario Generation → **Map Generation** → Goal Randomizer → ROS 2 Setup → ROS 2 Launch → Sim-ROS clock synchronization → Data Logger

Figure 7. Generated Warehouse Scenario

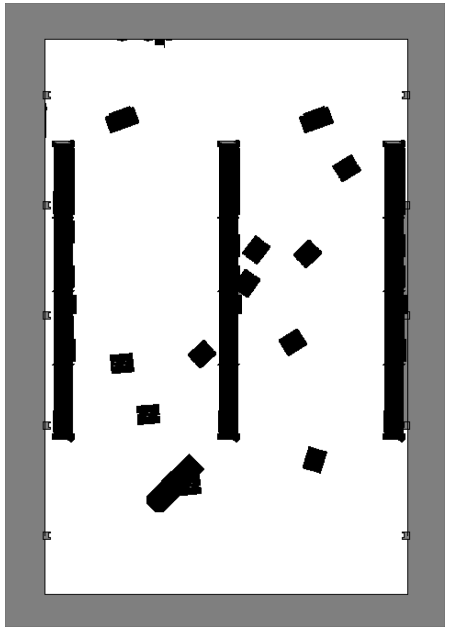
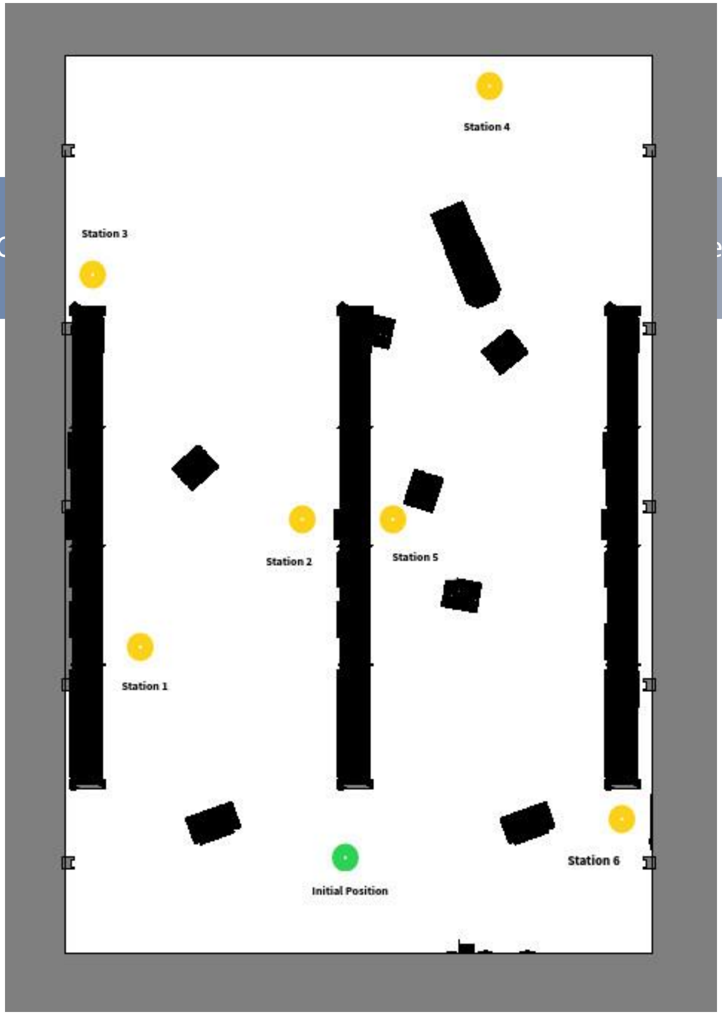


Figure 8. OGM of the scenario

Framework Development

Simulation Setup → Plausible Scenario Generation → Map Generation → Goal Randomizer → ROS 2 Setup → RC … er



Objective:

1. Accept a **list of goal** stations to visit as a task
2. Randomize the list for each simulation
3. Save the list as a file to be read by *Goal publisher* package as well as for future reference

Goal Station	Position	Orientation
#1	(px1, py1)	(ox1, oy1, oz1, ow1)
#2	(px2, py2)	…
#3	…	…
…	…	…

Framework Development

Simulation Setup → Plausible Scenario Generation → Map Generation → Goal Randomizer → ROS 2 Setup → ROS 2 Launch → Sim-ROS clock synchronization → Data Logger

Nav2 Setup (***carter_navigation*** package)

Robot Setup (***carter_description*** package)

Task Package (***pub_navigation_goal*** package)

Rosbag Recorder (***bag_recorder*** package)

Setup important ROS 2 packages

→ Launching ROS 2

Framework Development

Simulation Setup | Plausible Scenario Generation | Map Generation | Goal Randomizer | ROS 2 Setup | ROS 2 Launch | Sim-ROS clock synchronization | Data Logger

Objective:

- Ensure **time synchronization** between Isaac Sim and ROS 2.
- Configure an Isaac Sim action graph for ROS Clock synchronization using **OmniGraph nodes**.

OmniGraph is NVIDIA Isaac Sim's ***visual programming framework*** that enables the seamless connection of functions from various Omniverse (extension) systems through a graphical interface.

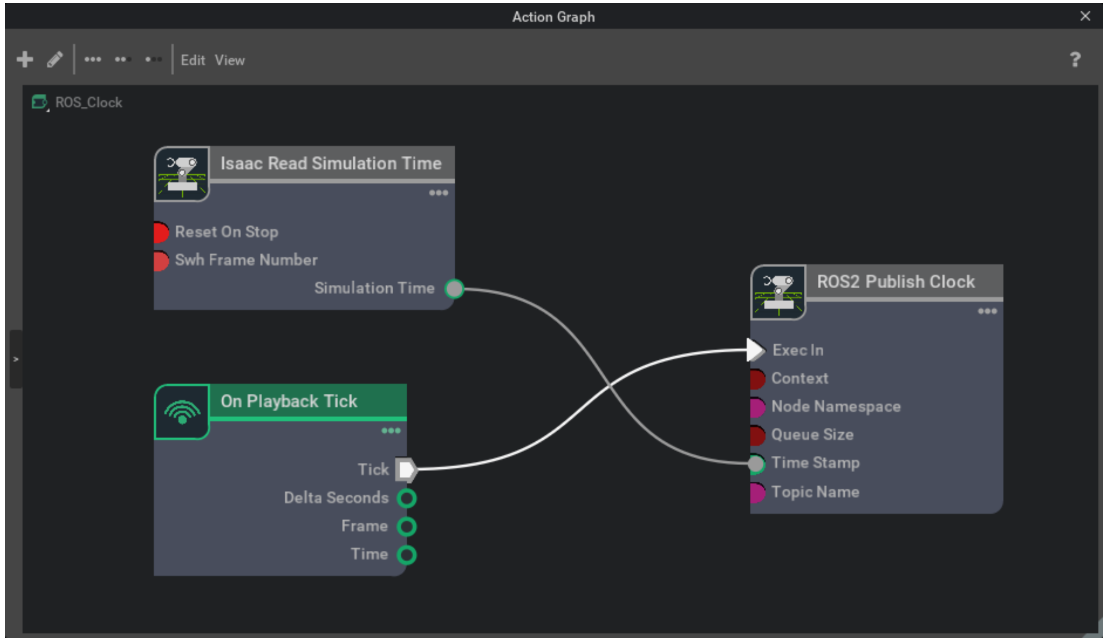


Figure 9. ROS_clock action graph for Sim-ROS clock synchronization

Framework Development

Table 1. Data recorded from mobile robot simulation

No.	Type	Fields	Description
1	Time and Time Step Information	current_time, current_time_step	Current simulation time and time step index.
2	Robot Pose and Orientation	px, py, pz, ox, oy, oz, ow	Robot position (x, y, z) and orientation.
3	Robot Velocity	vx, vy, wz	Linear and angular velocity of the robot.
4	Goal Status	goal_status	Indicates if the robot is approaching any goal station.
5	Motion Type Flags	idle, linear, rotational	Boolean flags indicating robot's motion type.
6	Navigation Behaviour Tree Status	Various flags representing behavior status	Status of actions within the behavior tree [SUCCESS/RUNNING/FAILURE].
7	Contact Sensor Boolean Flag	RobotBodyContact	Indicates if the robot made contact or collisions.

Framework File Structure

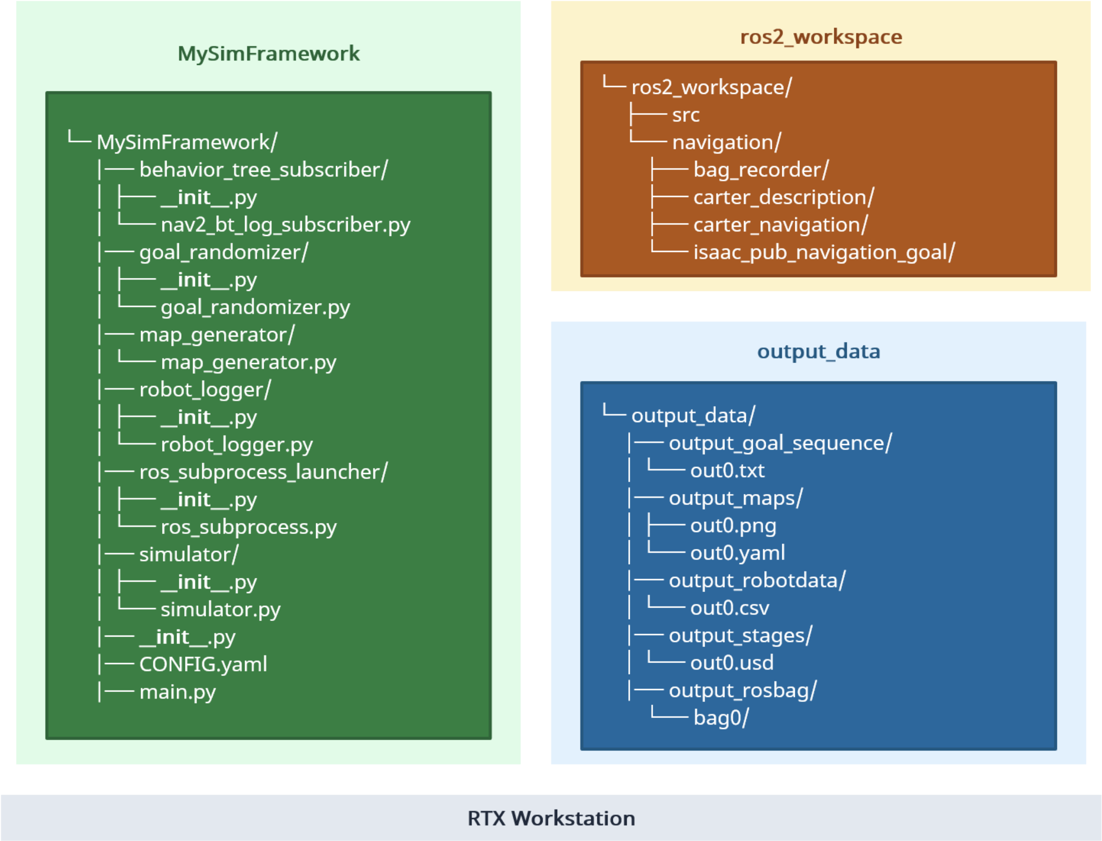
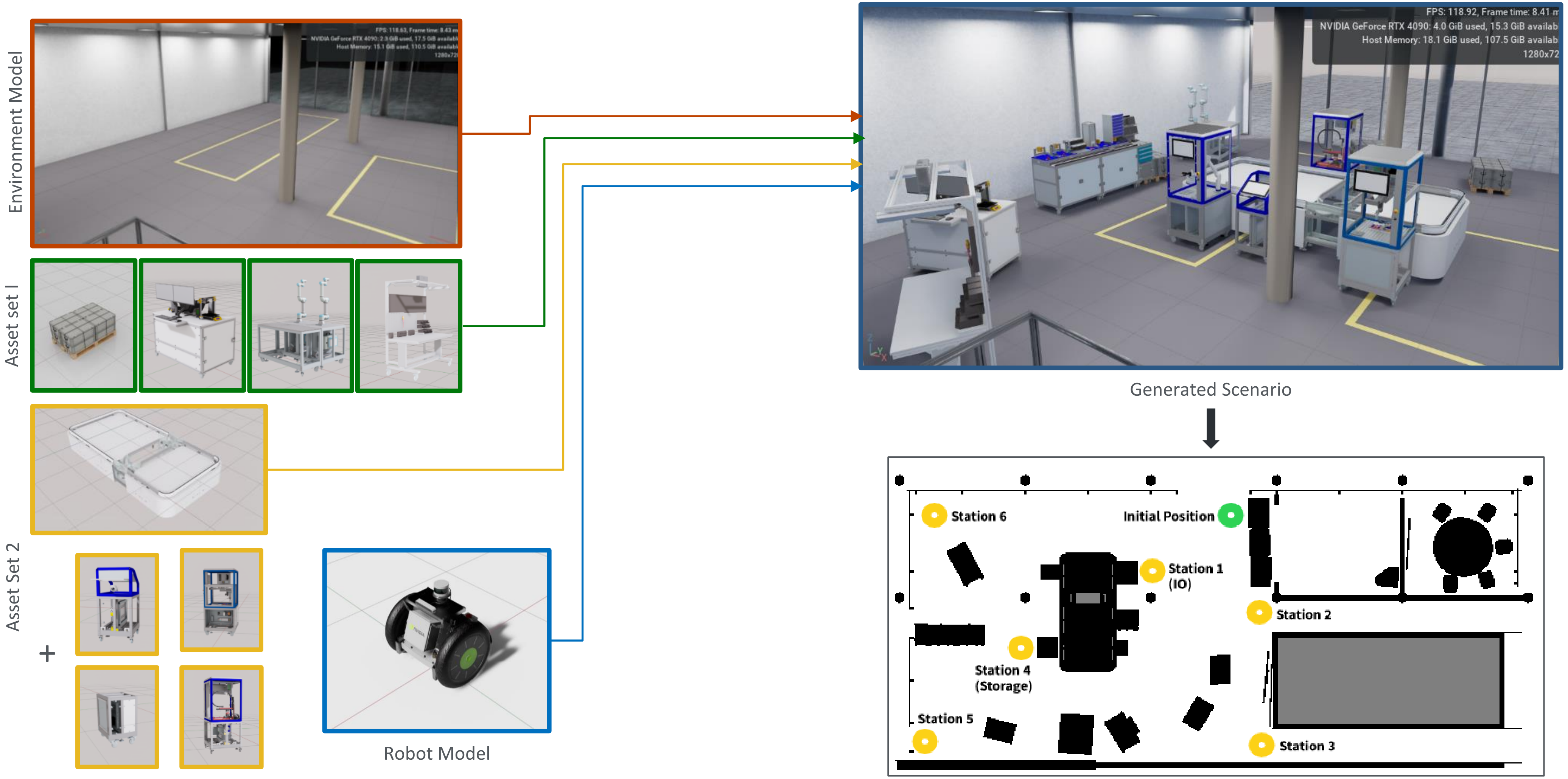


Figure 10. File Structure of the Framework

Scene and Taskset Generation



Generated Map and Goal Taskset for Robot

Mobile Robot Navigation Simulation in the Generated Scenario

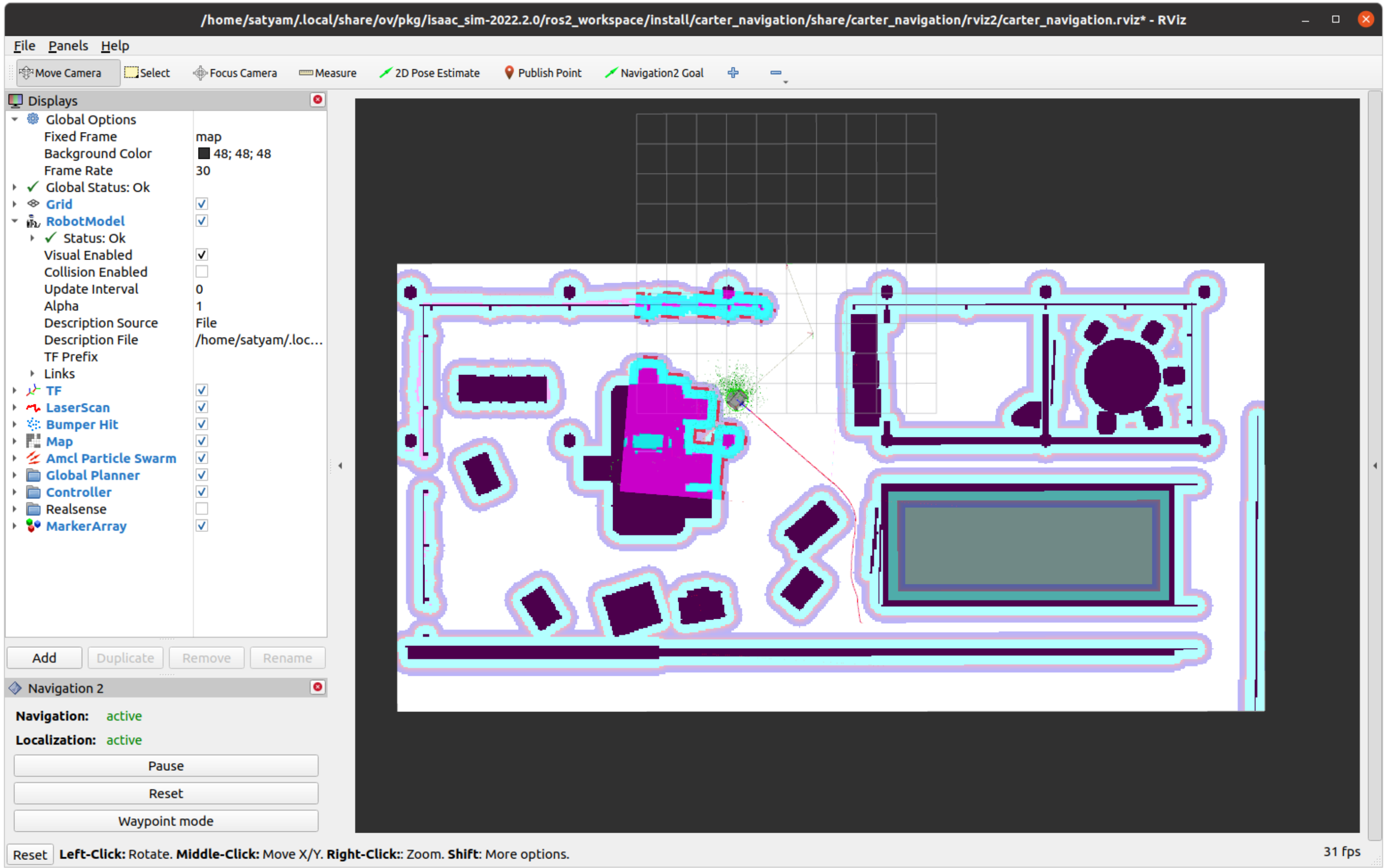


Figure 11. Navigation visualization in Rviz2

Generated Dataset

For each simulation,

- Objective Set (List of goal in **TXT file**)
- Generated Scenario Stage (**USD file** of the scene)
- Generated Occupancy Grid Map (**PNG and YAML map** data)
- Simulation data (**CSV file**)
- **Rosbag**

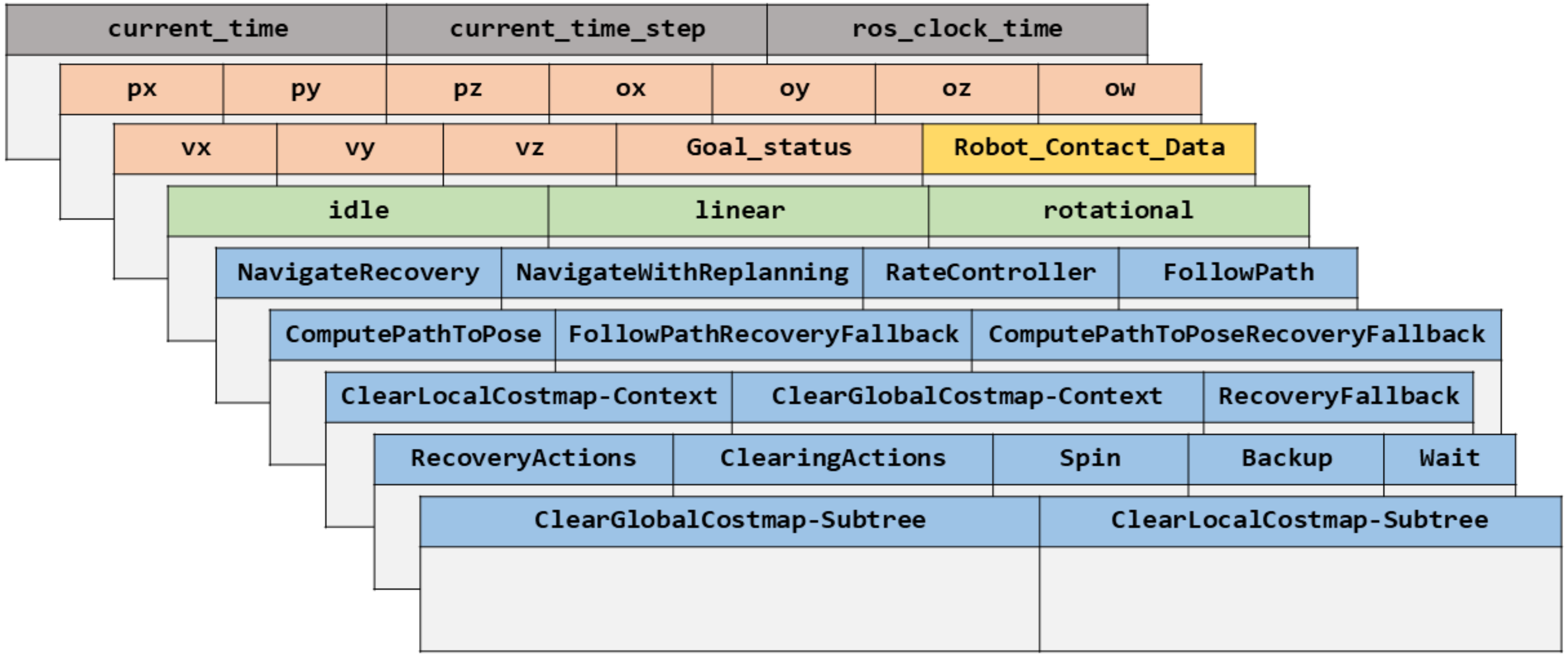


Figure 12. Simulation data CSV header names

Applications:

Application Areas:

- Robotic Testing and Optimization.
- Development of AI-based Robotic Solutions.
- Path Planning and Perception Models.
- Digital Twin Integration.

Notable Features:

- Digital Sensor and Actuator Validation.
- ROS Navigation System Integration.
- Custom Environment Configurations.
- Systematic Simulation Data Recording.

Future Work

Opportunities:

- Integration into CI/CD Pipelines.
- Synthetic Data Training Models.
- Continued Framework Refinement.
- AI-Based Robotic Integration Applications.
- Configure to support multiple robots or robot fleet.

Considerations:

- Framework Designed for Easy Maintenance.
- Potential Contribution to Digital Twins Development.

References:

[1] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems, 16, 3 2019. *ISSN 17298814. doi: 10.1177/1729881419839596*.

[2] Khaled El Emam. Accelerating ai with synthetic data generating data for ai projects. 2020.

[3] Maximiliano Rojas, Gabriel Hermosilla, Daniel Yunge, and Gonzalo Farias. An easy to use deep reinforcement learning library for ai mobile robots in isaac sim. Applied Sciences 2022, Vol. 12, Page 8429, 12:8429, 8 2022. *ISSN 2076-3417. doi: 10.3390/APP12178429.*

[4] Giuseppe Fragapane, René de Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. European Journal of Operational Research, 294:405–426, 10 2021. ISSN 0377-2217. doi: 10.1016/J.EJOR.2021.01.019.

[5] ukovicaleksa. Github - lukovicaleksa/grid-mapping-in-ros: Creating occupancy grid maps using static state bayes filter and bresenham's algorithm for mobile robot (turtlebot3burger) in ros. *URL https://github.com/lukovicaleksa/grid-mapping-in-ROS.*

[6] Open Navigation LLC. Nav2 — navigation 2 1.0.0 documentation. *URL https://navigation.ros.org/.*

[7] NVIDIA. Isaac sim - robotics simulation and synthetic data generation | nvidia developer. *URL https://developer.nvidia.com/isaac-sim.*

[8] NVIDIA. Included environments and robots — isaacsim latest documentation. *URL https: //docs.omniverse.nvidia.com/isaacsim/latest/reference_assets.html.*

[9] Daniel Michelon De Carli, Fernando Bevilacqua, Cesar Tadeu Pozzer, and Marcos Cordeiro d'Ornellas. A survey of procedural content generation techniques suitable to game development. IEEE, 11 2011. *ISBN 978-1-4673-0797-0. doi: 10.1109/SBGAMES.*

[10] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Advances in Neural Information Processing Systems, 2016.

[11] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d-scene generation with neural radiance fields. 5 2023.

[12] Yu Cheng, Zhiyong Sun, Yan Shi, and Lixin Dong. Controllable scene generation from natural language. Procedia Computer Science, 209:122–131, 1 2022. *ISSN 18770509. doi: 10. 1016/j.procs.2022.10.106.*

Questions?

Thank You for attending!

BACKUP

Significance of the Research and Development

Solution to the key challenge: Autonomously Generating vast, reliable, and efficient datasets for robotics applications.

Innovation in Simulation: Advancement in robotics simulation to meet data needs.

Construction of 3D Digital Twins: Play important role for accurate environment representations for AMRs.

Allow Scalable Navigation Testing: Quickly simulated testing within varied robot navigation and environments factors.

NVIDIA Omniverse as a Solution: Introduction to NVIDIA Omniverse as a potential solution for development of customized simulation framework or even a pipeline.

Mobile Robots and Core Components

- Definition: Autonomous or semi-autonomous machines capable of navigating various environments.
- Significance: Versatile applications across industries due to advances in robotics, AI, and sensors.
- Evolution: Progress from fixed industrial robots to mobile robots driven by sensor tech and navigation algorithms.
- Mobility Types: Various locomotion techniques, including wheeled, tracked, legged, airborne, and aquatic systems, have been explored.
- Core Components: Locomotion, Perception, Cognition, and Navigation.

Figure 1. Mobile robot types based on locomotion technique: Differential (top); Holonomic (middle); and Legged (bottom) [xx]

Nav2 Behavior Tree

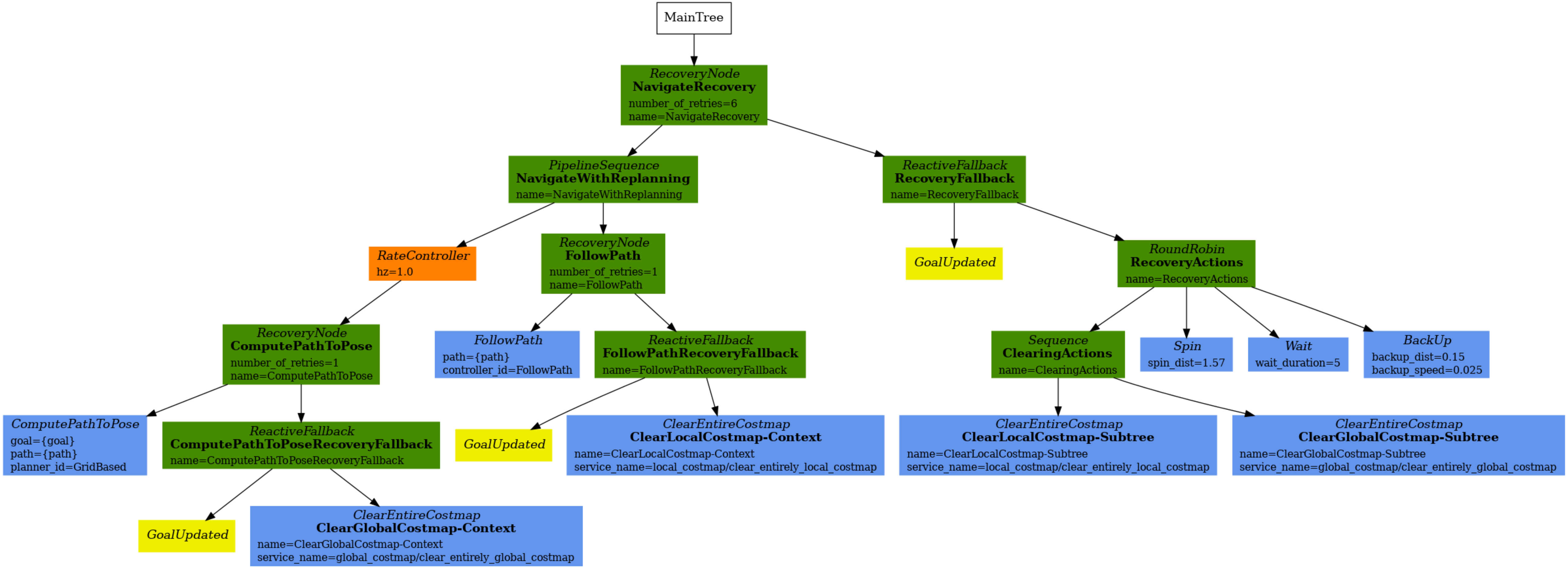
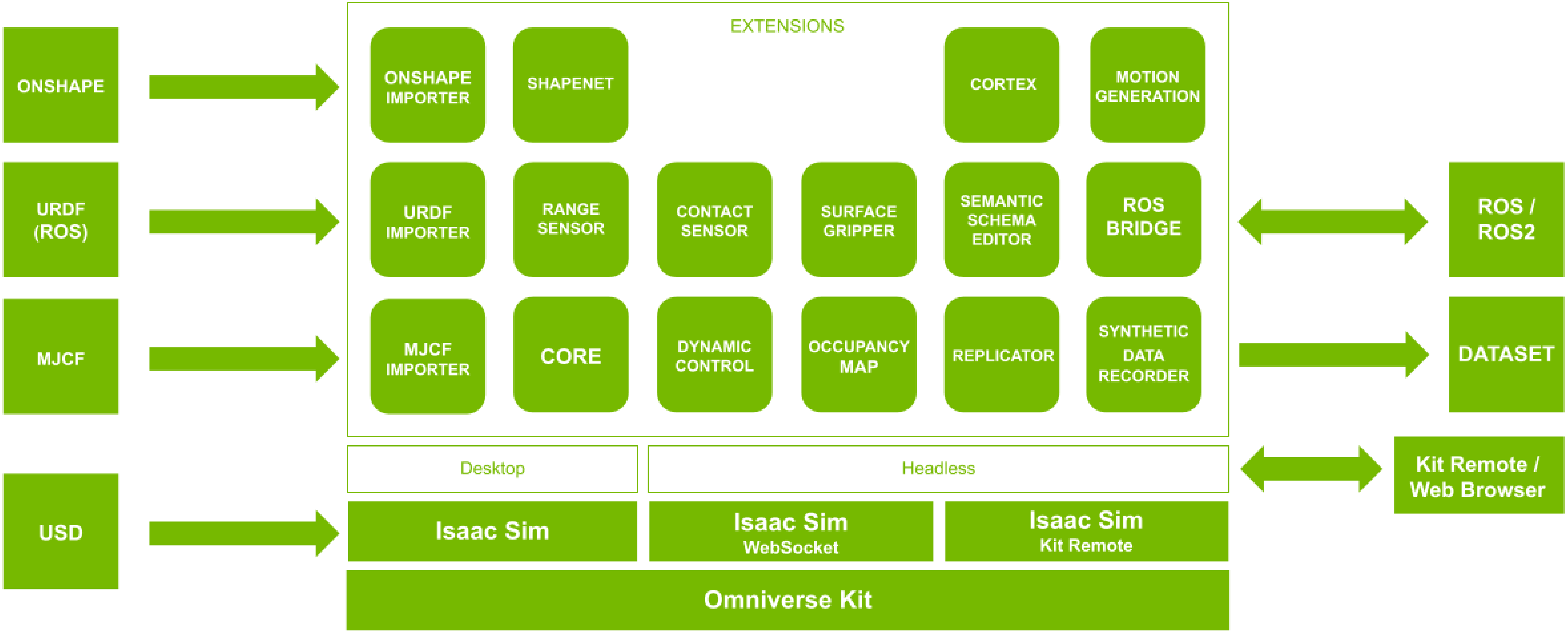
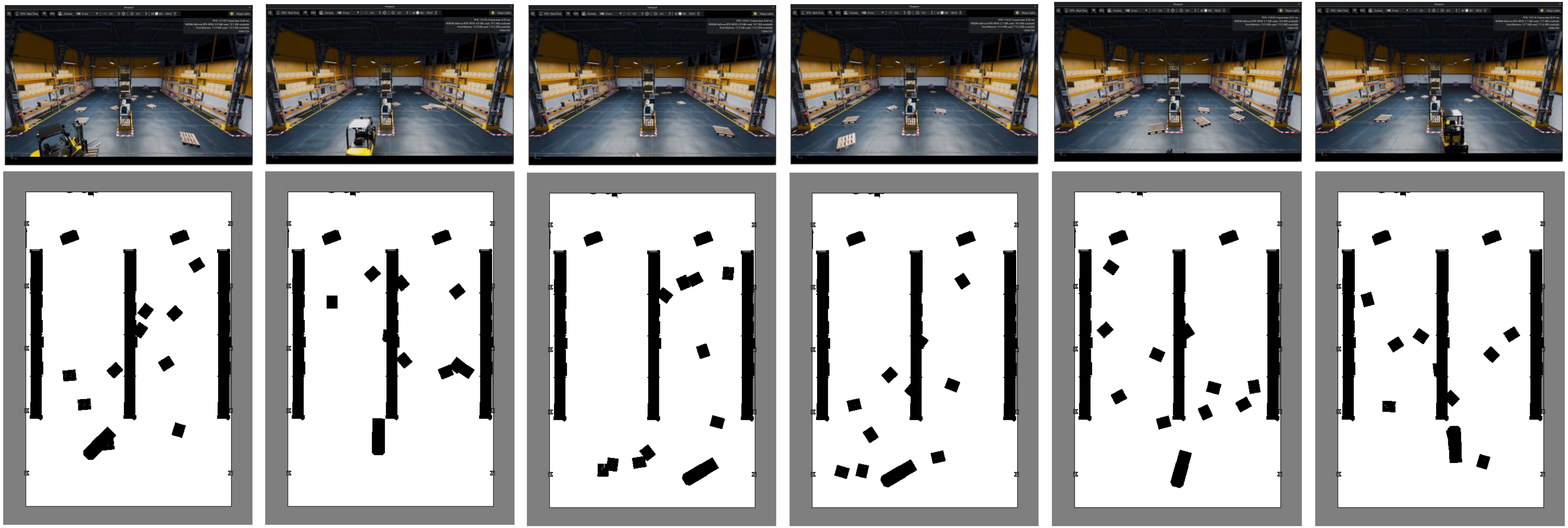


Figure 11. Navigate to Pose with Replanning and Recovery [5]

Extensions Isaac Sim



Warehouse Datasets



SmartFactory Lab Datasets