Contents

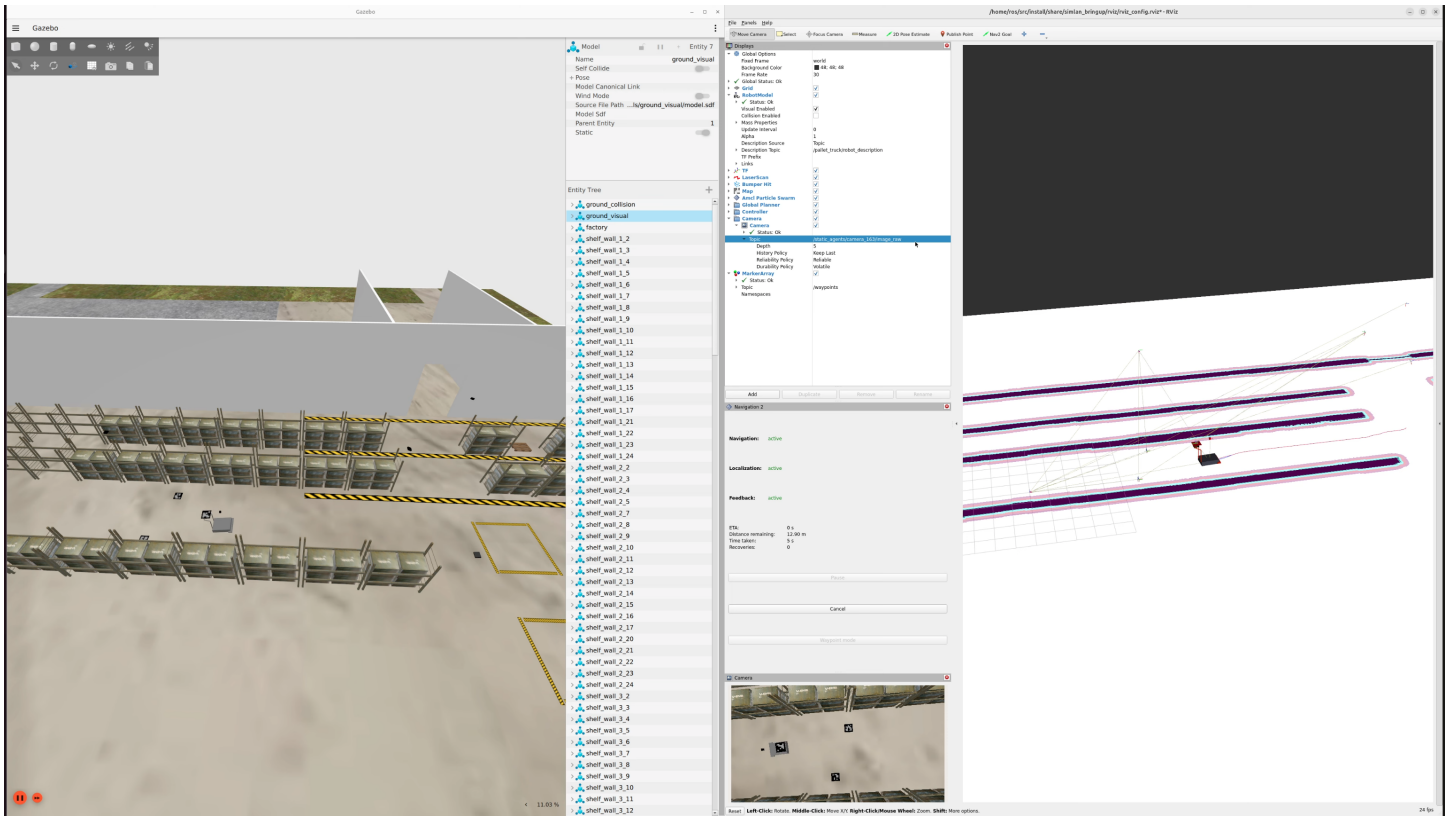
1 SIMLAN, Simulation for Multi-Camera Robotics (3.0.0)

This simulation environment, based on the Gazebo Ignition simulator and ROS2, resembles a Volvo trucks warehouse and serves as a playground for rapid prototyping and testing of systems that rely on multi-camera setup for perception, monitoring, localization or even navigation. This project is inspired by GPSS (Generic photo-based sensor system) that utilizes ceiling mounted cameras, deep learning and computer vision algorithms, and very simple transport robots. [GPSS demo]

1.1 SIMLAN Features [SIMLAN demo]

- Ignition Gazebo
- Library of assets
- Real-World environment inspired design (camera position and warehouse layout)
- ROS 2 interfaces (Humble and Jazzy)
- ArUco marker localization
- Simple GPSS (Generic Photo-based Sensor System) navigation
- Multi-Robot localization and navigation using Nav2
- Bird's-Eye view projection
- Multi-Sensor Support (LiDAR, RGB camera, semantic segmentation, Depth etc.)
- Geofencing for safe zones and safestop on collision
- Motion capture for Human-Robot Collaboration/Interaction (HRC/HRI)

Please click the youTube link below to view the SIMLAN demo video:



Here are list of advantages of using SIMLAN Multi-Camera system

- Rapid prototyping and iteration of ML based algorithm. (e.g. reinforcement learning)
- Enhanced monitoring and coordination using bird eye view
- Simplified robot design and maintenance.
- Extendible with additional ML based vision systems
- Safety testing without physical risk or privacy concerns
- Scalable and reproducible testing (CI/CD)
- Cost-effective development

1.2 Installation [Demo]

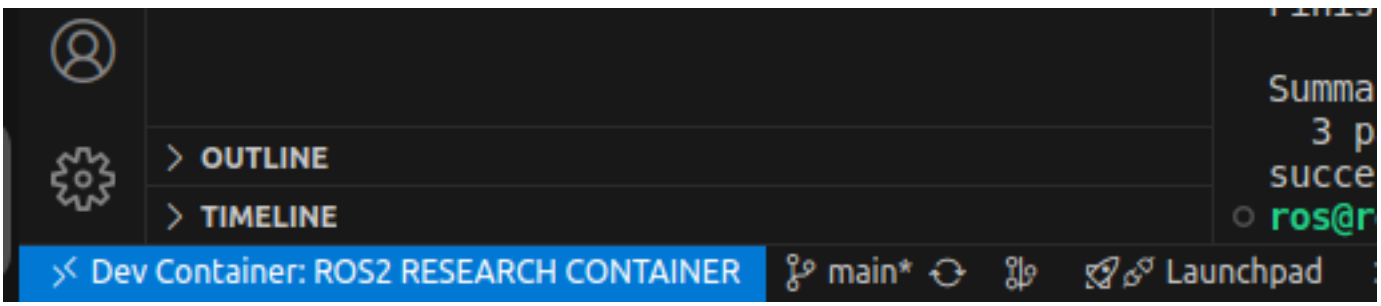
1.2.1 Dependencies

Ubuntu 24.04: use instruction in dependencies.md#linux-dependencies to install docker and ensure that your linux user account has `docker` access. *Attention*: Make sure to restart the computer (for the changes in group membership to take effect.) before proceeding to the next step.

Windows 11: use instruction in dependencies.md#windows-dependencies to install dependencies.

Production environment: follow installation procedure used in .devcontainer/Dockerfile to install dependencies.

Development environment: to improve collaboration we use vscode and docker as explained in this instruction and docker files. Install Visual Studio Code (VS Code) and open the project folder. VS Code will prompt you to install the required extension dependencies. Make sure the `Dev containers` extension is installed. Reopen the project in VS Code, and you will be prompted to rebuild the container. Accept the prompt, this process may take a few minutes. Once VS Code is connected to Docker (as shown in the image below), open the terminal and run the following commands:



(if you don't see this try to build manually in vscode by pressing `Ctrl + Shift + P` and select `Dev containers: Rebuild and Reopen in container`.)

1.2.2 Quick Start

The best place to learn about the various features, start different components, and understand the project structure is `./control.sh`.

Attention: The following commands (using `./control.sh`) are executed in a separate terminal tab inside *vscode*.

To kill all the relevant process (related to gazebo, ros2), delete build files, delete recorded images and rosbag files using the following command:

```
./control.sh clean
```

To clean up and build the ros2 simulation

```
./control.sh build
```

(optionally, in vscode you can click on Terminal-> Run Task/Run build Task or use `Ctrl + Shift + B`)

1.3 GPSS controls (pallet trucks, aruco) [Demo]

It is possible for the cameras to detect ArUco markers on the floor and publish their location to TF, both relative to the camera, and the ArUcos transform from origin. The package ./camera_utility/aruco_localization contain the code for handling ArUco detection.

You can also use nav2 to make a robot_agent (that can be either robot/pallet_truck) navigate by itself to a goal position. You can find the code in simulation/pallet_truck/pallet_truck_navigation

Run these three in separate terminals

```
./control.sh gpss # spawn the simulation, robot_agents and GPSS ArUco detection
./control.sh nav  # spawn map server, and separate nav2 stack in a separate namespace for each robot_agent
./control.sh send_goal # send navigation goals to nav2 stack for each robot_agent
```

If you want to control any robot (pallet truck, humanoid, etc) manually you can run the following command. Remember to specify what robot you want to control by adding its namespace as argument, i.e. `./control.sh teleop pallet_truck_1`

```
./control.sh teleop ${YOUR_ROBOT_NAMESPACE}
```

If you want to record any of your topics during the tests you can run the following command. Change the topic in the control.sh script: `ros2 bag record /topic` to whatever topic you want to record.

```
./control.sh ros_record
```

To replay your latest recorded rosbag run the following command:

```
./control.sh ros_replay
```

If you want to do a camera dump and save the image from each camera as a .png run the following command. The images will appear at `/src/camera_utility/camera_number`.

```
./control.sh camera_dump
```

If you want to take a screenshot of one of the cameras view, run the following command. Replace `###` with the camera you want to take a screenshot of. (163, 164, 165 or 166)

```
./control.sh screenshot ###
```

```
./control.sh birdeye
```

If you want to add the tf links between the cameras and the ArUco markers without running the `gpss` command you can run the following command. This is not that usable as the `gpss` run this as well, but it can be good for debugging.

```
./control.sh aruco_detection
```

Finally, to view the bird's-eye perspective from each camera, run the following command and open `rviz` Then, navigate to the scroll menu to the left, and under "Camera" change the Topic `/static_agents/camera_XXX/image_projected` topic to visualize the corresponding camera feed:

1.4 RITA controls (humanoid, robotic arm) [Demo]

```
./control.sh humanoid
```

To move humanoid around in the simulator

```
./control.sh teleop ${YOUR_HUMANOID_NAMESPACE}
```

1.4.0.1 Arm controls Spawn the Panda arm inside SIMLAN and instruct it to pick and place a box around with the following commands:

```
./control panda
./control plan_motion
./control pick
```

1.5 Advanced options

See resources/ISSUES.md to learn about additional advanced options and to check known issues before reporting any issue or requesting new features. To start the project **without NVIDIA GPU** please comment out these lines in `docker-compose.yaml` as shown below:

```
  #   runtime: nvidia
  #
  # factory_simulation_nvidia:
  #  <<: *research-base
  #  container_name: factory_simulation_nvidia
  #  runtime: nvidia
  #  deploy:
  #    resources:
  #      reservations:
  #        devices:
  #          - driver: nvidia
  #            count: "all"
  #            capabilities: [compute,utility,graphics,display]
```

`camera_enabled_ids` specifies which cameras are enabled in the scene for ArUco code detection and birdeye view.

1.5.1 Customized startup

In `config.sh` it is possible to customize your scenarios. From there you can edit what world you want to run, how many cameras you want enabled, and also edit Humanoid related properties. Modifying these variables are preferred, rather than modifying the `control.sh` file.

1.5.2 World fidelity

in the `config.sh` script, you can adjust the world fidelity

The active worlds are:

arguments	configuration
`default`	Contains the default world with maximum objects
`medium`	Based on default but boxes are removed
`light`	Based on medium but shelves are removed
`empty`	Everything except the ground is removed

1.5.3 Filtering log output

In `config.sh` you can set the level of logs you want outputted into the terminal. Per default it is set to “info” to allow all logs. Possible values are: “debug”, “info”, “warn”, and “error”. Setting it to “warn” filters out all debug and info messages. Additionally, to filter out specific lines you can add the phrase you want filtered, inside of `log_blacklist.txt` and setting the `log_level` flag to “warn” or “error” will start filtering out all phrases found in the blacklist.

1.5.4 Older versions

- `gz_classic_humble` branch contain code for **Gazebo Classic (Gazebo11)** that has reached end-of-life (EOL).
- `ign_humble` branch contain code for **ROS2 humble & Gazebo ignition**, an earlier version of this repository.

1.6 Documentation

Learn more about the project by reading these documents:

- `control.sh` script is a shortcut to run different launch scripts, please also see these diagram.
- `config.sh` contains information about, which world is loaded, which cameras are active, what and where the robots are spawned.
- Marp Markdown Presentation
- Pallet Truck Navigation Documentation
- Camera Utilities and notebooks: (Extrinsic/Intrinsic calibrations and Projection)
- Humanoid Utilities (pose2motion)
- Configuration Generation
- `simulation/`: ROS2 packages
  - Simulation and Warehouse Specification (fidelity)
  - Building Gazebo models (Blender/Phobos)
  - Objects Specifications
  - Warehouse Specification
  - Aruco Localization Documentation
  - humanoid_robot Simulation
  - Geofencing and Collision safe stop
  - Visualize Real Data **requires data from Volvo**
  - Humanoid Control
- `CHANGELOG.md`
- `credits.md`
- `LICENSE` (apache 2)
- `contributing.md`

1.7 Research Funding

This work was carried out within these research projects:

- The SMILE IV project financed by Vinnova, FFI, Fordonsstrategisk forskning och innovation under the grant number 2023-00789.
- The EUREKA ITEA4 ArtWork - The smart and connected worker financed by Vinnova under the grant number 2023-00970.

INFOTIV AB	Dyno-robotics	RISE Research Institutes of Sweden	CHALMERS	Volvo Group

SIMLAN project is started and is currently maintained by Hamid Ebadi. To see a complete list of contributors see the changelog.

1.7.1 3.1.0: Improvements (Dec 2025) - Supervisor: Hamid Ebadi

- System integration tests by Pär
- System diagram by Anton
- Pytorch model by Pär
- Improved doc and AutoGluon configuration by Marwa

1.7.2 3.0.0: Jazzy, humanoid (Oct 2025) - Supervisor: Hamid Ebadi

- Collision sensor: Anton Stigemyr Hill
- Automatically generated parameter-files for robot_agents and world fidelity : Anton Stigemyr Hill
- Shared map and obstacle generation for robot_agents: Anton Stigemyr Hill
- Geofencing and safety stop: David Espedalen
- Humanoid motion capture: Casparsson and Siyu Yi, Hamid Ebadi
- Humanoid integration (pymoveit2 to moveit_py): Siyu Yi, Pär Aronsson
- Mocap (humanoid) with multi-camera training pipeline: Siyu Yi, Pär Aronsson
- Panda arm integration: Pär Aronsson
- Dyno-robotics trajectory replay integration, scenario: Sebastian Olsson

1.7.3 2.1.0: Namespace cleanup and multi-agent navigation (Aug 2025) - Supervisor: Hamid Ebadi

- Implemented generic robot-agent control supporting multiple meshes (e.g., pallet truck, forklift) and distinct ArUco IDs: Pär Aronsson
- Robot-agent and navigation2 (nav2) namespace: Pär Aronsson
- Multi-agent navigation capabilities using GPSS: Pär Aronsson

1.7.4 2.0.0: GPSS (May 2025) - Supervisor: Hamid Ebadi

- Upgrade Gazebo from classic to ignition, adapting new sensors: Nazeeh Alhosary
- Aruco_localization, added localization and nav2 to new robot: Pär Aronsson
- Navigation: Pär Aronsson
- Bird's Eye View ros package: Converting projection.ipynb to camera_bird_eye_view: Hamid Ebadi, Pär Aronsson
- Deprecated: Infobot, replaced with Pallet_truck (based on Jackal by Clearpath Robotics)

1.7.5 1.0.6: Collision (Feb 2025)

- Add scenario execution library for ros2
- Add action servers for set_speed, teleport_robot and collision
- Add Node for TTC calculation
- Add package for custom messages

1.7.6 1.0.5: Delivery 4 (Dec 2024)

- D3.5 Disentanglement: One-parameter Object Movements.
- Trajectory visualization : Hamid Ebadi
- feat: added DynoWaitFor to make sure Jackal is launched last (synchronisation issue)
- simlan_bringup pkg created : Christoffer Johannesson

1.7.7 1.0.4: Delivery 3 (Oct 2024)

- SMILE-IV and ArtWork projects contributions
- Updated camera extrinsic, fixing OpenCV camera calibration to Gazebo simulator conversion : Erik Brorsson, Volvo
- D3.4 Out distribution data collection : Hamid Ebadi
- D3.3 In distribution data collection : Hamid Ebadi
- D3.2 Images for stitching : Hamid Ebadi
- D3.1 Dataset draft for HH : Hamid Ebadi
- CI/CD and Kubernetes integration by Filip Melberg, Vasiliki Kostara
- Jackal integration : Christoffer Johannesson, Hjalmar Ruscck
- Docker/vscode

- Disable GPU support by default
- POC for camera projection to pixel coordinates (jupyter notebook) : Hamid Ebadi
- Camera image dump and image stitching : Hamid Ebadi

1.7.8 1.0.3: Jackal Robot (Mar 2024) - Christoffer Johannesson

- Added dyno fork of jackal repo from Clearpath Robotics.
- Updated to Humble, added bringup and support for namespacing. Jackal can be spawned in Gazebo and controlled through the keyboard.
- Added .devcontainer folder with Dockerfile and devcontainer.json to set up project container in VS Code.
- Added docker-compose to link all needed files and set environment variables.
- Added .vscode folder with settings and tasks for easy building of the project.
- Updated README with info on how to use Docker setup in VS Code, and some features to make it easy to share the same setup with others.
- Features includes: python3 dependency install with pip, cloning of other git repositories and how to make changes to those repositories.

1.7.9 1.0.2: Delivery 2 (Feb 2024)

- Volvo warehouse 0.0.1 : Hamid Ebadi
- Volvo camera calibration in Gazebo 0.0.1 : Hamid Ebadi
- Integrate Infobot_agent 0.0.2: InfoBot differential-drive AMR (Autonomous Mobile Robot) URDF and ROS launcher (GOPAL and forklift): Hamid Ebadi
- Integrate Infobot_cartographer 2.1.5: cartographer for creating PGM maps
- Integrate nav2_commander 0.0.2: ROS package to command Infobot where the destination is : Hamid Ebadi
- Integrate Infobot_navigation2 2.1.5: Standard Nav2 stack launcher : Hamid Ebadi
- Integrate Infobot_teleop 0.0.2: Teleoperation for InfotBot

1.7.10 1.0.1: Delivery 1 (Dec 2023) - Supervisor: Hamid Ebadi

- Basic warehouse model 1.0.0: Anders Bäckelie
- CAD modelling (eur-pallet, boxes, shelf, support_pole, traffic-cone, steel_drum) 1.0.0 : Jacob Rohdin
- Physics (collision, inertia), visuals and Gazebo compatible mesh creation 1.0.0: Anders Bäckelie
- Walking actor using scripted trajectories 1.0.0 : Anders Bäckelie
- Infobot_Gazebo_environment 1.0.0: ROS2 launcher to start Gazebo world : Hamid Ebadi
- static_agent_launcher 1.0.0: Camera and Aruco tags : Hamid Ebadi
- camera-viewer 1.0.0: Python code to get Gazebo camera feed : Hamid Ebadi