Repository README

Coordinated Robot Search

Bachelor Project 2025

- Thesis Report
- botbrain documentation
- simple_sim documentation

`botbrain`

Rust library crate containing the core logic of the project including the search algorithms.

Documentation can be found here.

`simple_sim`

A simple simulator for running behaviors defined in `botbrain`.

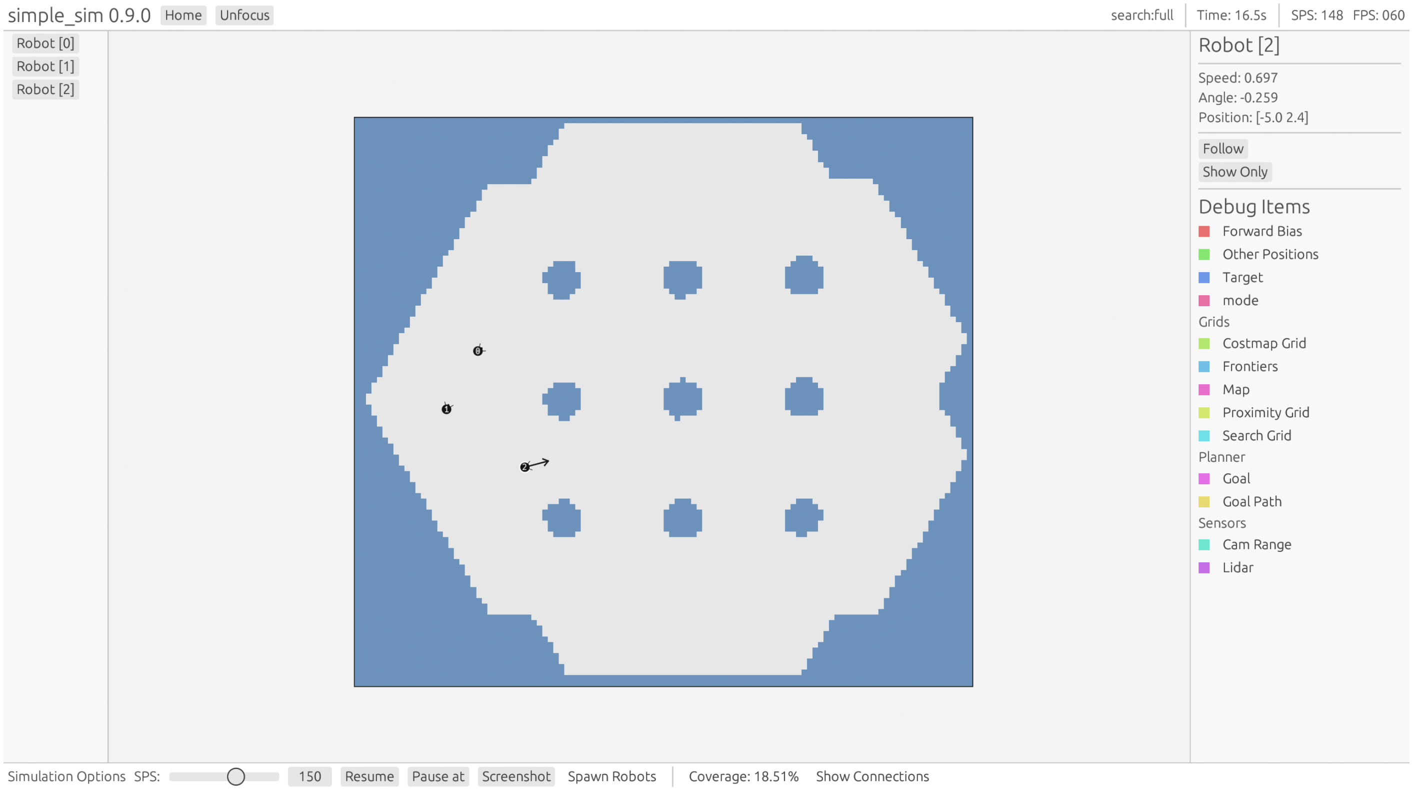
Start the simulation with:

```
cd simple_sim
cargo run --release -- run ../worlds/bitmap/depot/depot.yaml search:pathing
```

When the simulator opens, click `Spawn Robots` and click on the map to spawn robots. The speed of the simulation can be adjusted by dragging the slider in the bottom left. Information in the `Debug Soup` can be visualized by clicking on the robot and choosing a `Debug Item` in the right panel.

More launch options can be found with

```
cargo run --release -- --help
```



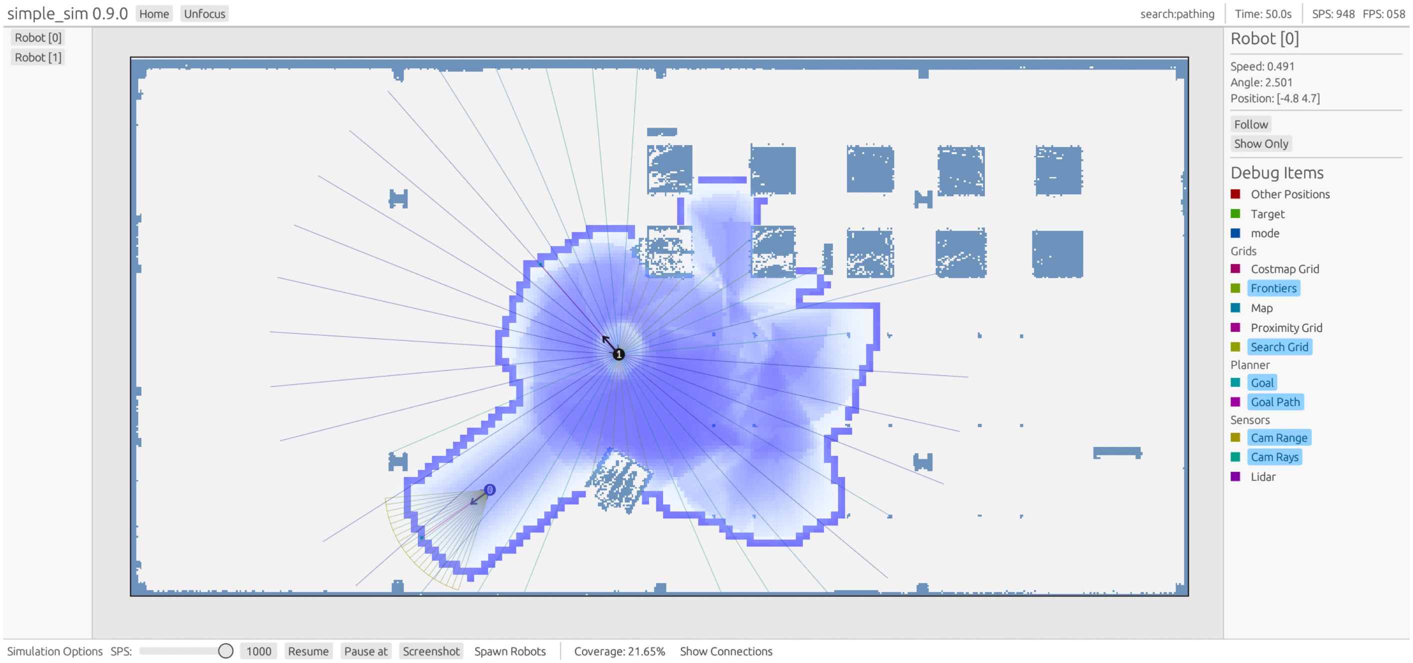
Time: 16.5s | SPS: 148 FPS: 060

Robot [2]

Speed: 0.697
Angle: -0.259
Position: [-5.0 2.4]

SPS: 150

Coverage: 18.51%



Time: 50.0s | SPS: 948 FPS: 058

Robot [0]

Speed: 0.491
Angle: 2.501
Position: [-4.8 4.7]

SPS: 1000

Coverage: 21.65%

ROS 2

The `multi_robot_control` ROS 2 package, contains the nodes needed to run the robot behaviors in ROS 2. `ros_agent` is the main behavior node which manages the botbrain robot state.

The ROS 2 packages can be built with:

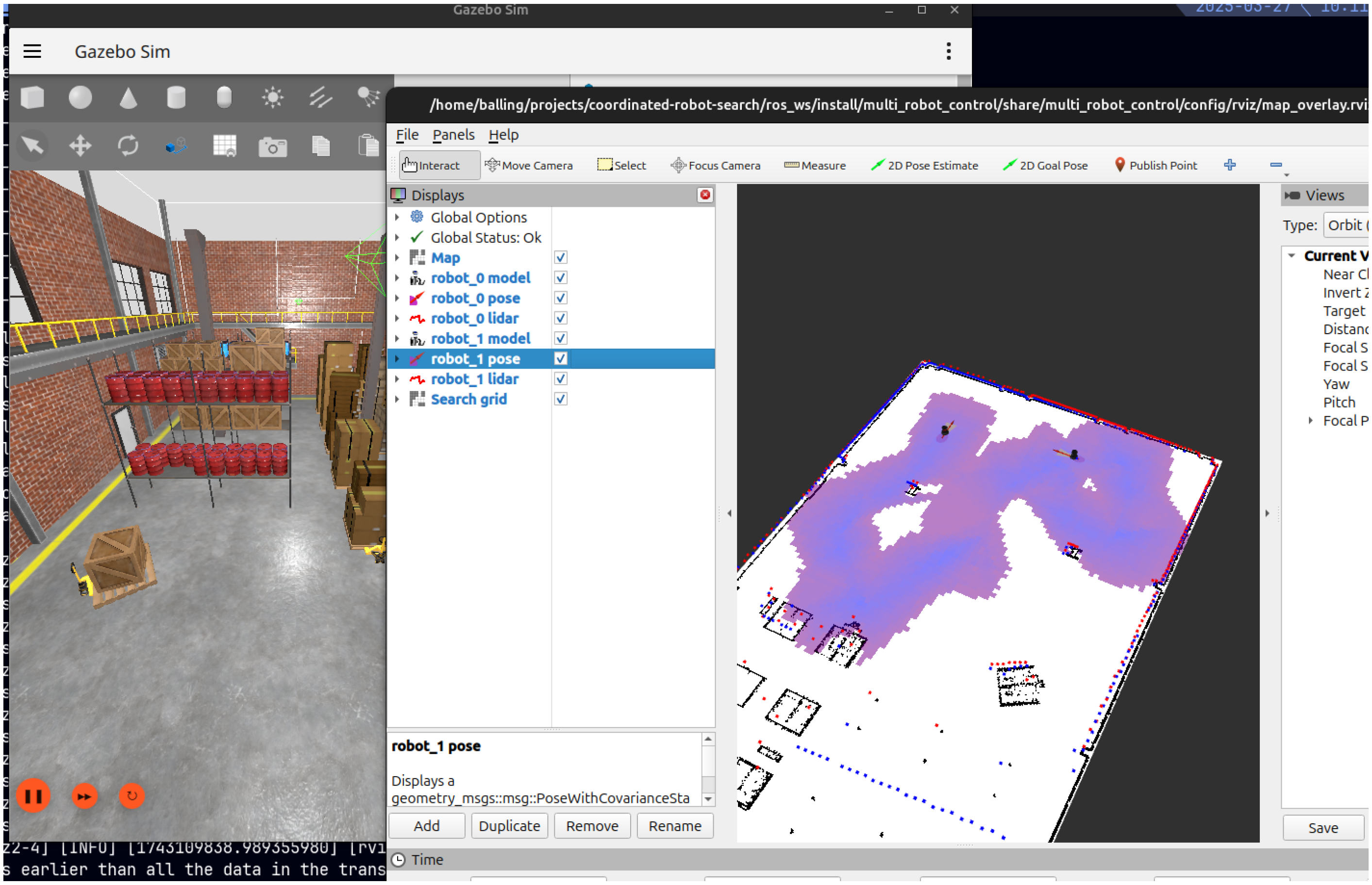
```
source /opt/ros/jazzy/setup.bash
cd ros_ws
colcon build --symlink-install
source install/setup.bash
```

And run with:

```
ros2 launch multi_robot_control multi_robot.launch.py behavior:=search robots:=0,0,0:2,0,1 map:=../w
```

This will launch Gazebo and Rviz2 windows, where the simulation can be observed.

See ROS 2 Agent for more information.



/home/balling/projects/coordinated-robot-search/ros_ws/install/multi_robot_control/share/multi_robot_control/config/rviz/map_overlay.rvi

```
z2-4] [INFO] [1743109838.989355980] [rvi
s earlier than all the data in the trans
```

robot_1 pose

Displays a geometry_msgs::msg::PoseWithCovarianceSta

Dockerfile A Dockerfile is provided to run ROS 2 and Gazebo. The provided `enter-docker.sh` can be used to build and enter the container.

```
./enter-docker.sh
```

To rebuild the container before running run

```
./enter-docker.sh --rebuild
```

Trainer

The `trainer` crate can be used to train models defined in `botbrain`. It defines the training loop and implements deep reinforcement learning.

See the trainer README for more information.

Plotting

The `botplot` python library is used to run `simple_sim` and Gazebo in an automated manner to collect data, which can be plotted with the provided functions. Simulator runs are cached to avoid rerunning scenarios. The `plotting` directory contains scripts which use `botplot` to run the simulators and create plots.

Using `botplot`

Install the package

```
python3 -m venv venv      # Create virtual enviornment
source venv/bin/activate  # Activate the enviornment
pip install -e ./botplot  # Install the `botplot` package
```

Run a plotting script

```
python3 ./plotting/report/sim_consistency.py
```

Compiling the LaTeX Report

```
cd report
latexmk -pdf -bibtex-cond -shell-escape -interaction=nonstopmode main.tex
```

Nix Integration

This project contains a `flake.nix` which allows self-contained build recipes and development shells. All dependencies are automatically downloaded.

Build the Project Build the entire project: - Executables: `simple_sim` and `trainer` - Documentation: `botbrain` and `simple_sim` - LaTeX Report

```
nix build .
```

The output is located in the `result` directory.

Sub-packages can be listed with the following

```
nix flake show
```

Development Shell Use the following command to install dependencies and start the development shell. ROS 2 is not supported with this shell, the `Dockerfile` can be used instead.

```
nix develop
```