mr_goto

Usage

Launch Map

ros2 launch stage_ros2 stage.launch.py

Start GoTo

ros2 run mr_goto goto --ros-args -p mode:="plan" -p x:=-3.0 -p y:=-5.0 -p deg:=70.0 --remap scan:=base_scan

Launch GoTo

ros2 launch mr_goto goto.launch.py

Launch Everything Everywhere All at Once

ros2 launch mr_goto full.launch.py

Call explored Area Service

ros2 service call /explored_area std_srvs/srv/Trigger

Installation

make build-ws02
restart terminal

Documentation

0 Documentation

0.1 Points

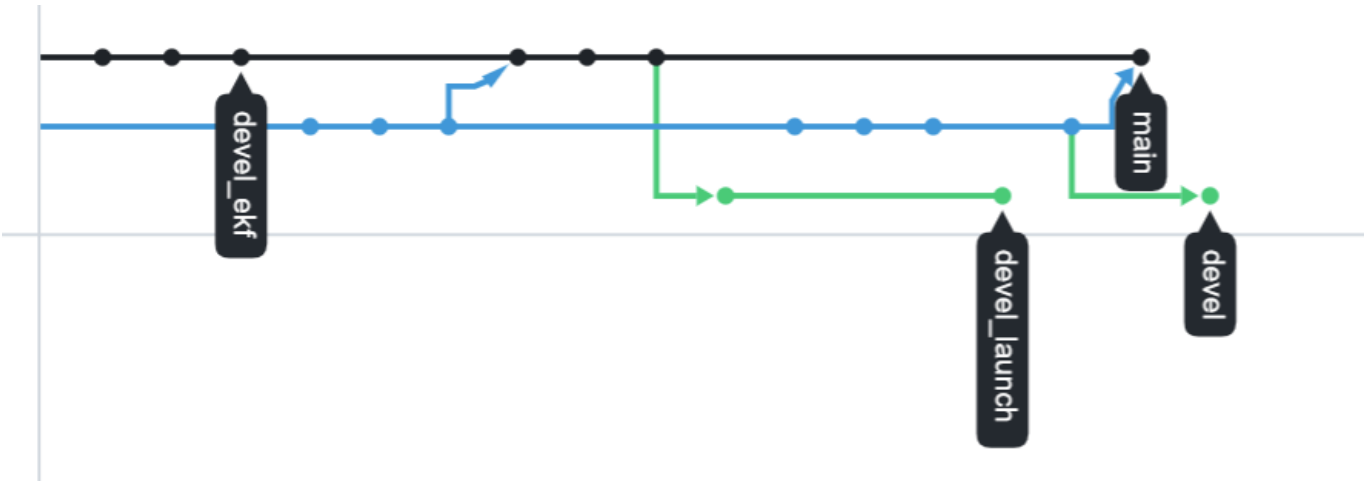
Task	Points	Author
0.	100%	All
0.2	20	All
1.1	50	Florian Wimmer
1.2.1	25	Simon Roth
1.2.2	25	Florian Wimmer
1.3.1	25	Simon Roth

Task	Points	Author
1.3.2	25	Simon Roth
5.1	20	Florian Wimmer
5.2	20	Florian Wimmer
5.3	20	Florian Wimmer
5.4	20	Florian Wimmer
6	50	Rita Schrabauer, Florian Wimmer
7.1	30	Simon Roth
7.2	20	Simon Roth
7.3	10	Simon Roth
Sum	**360**	**All**

Authors	Points(%)
Simon Roth	33.3
Rita Schrabauer	33.3
Florian Wimmer	33.3

0.2. Git usage

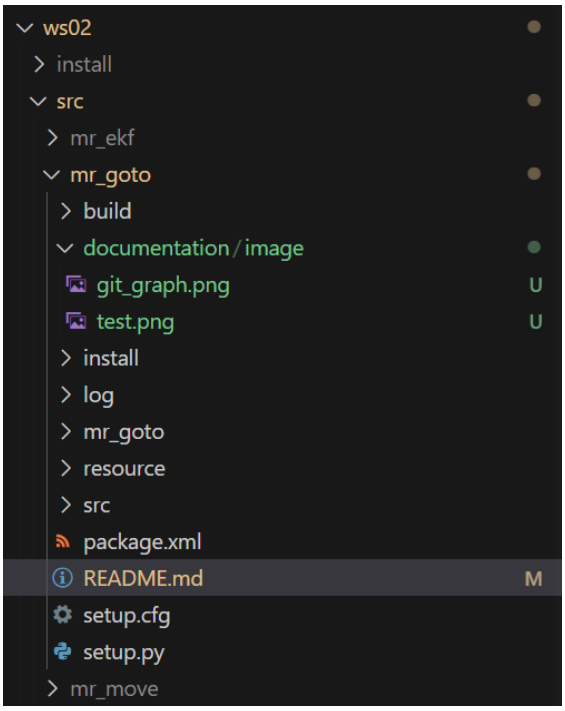
Git was used during the project, the ropsitory can be found under the following link: https://github.com/srothh/mr_goto



1 GoTo

1.1 New

The GoTo Task was implemented in a new Python Node.



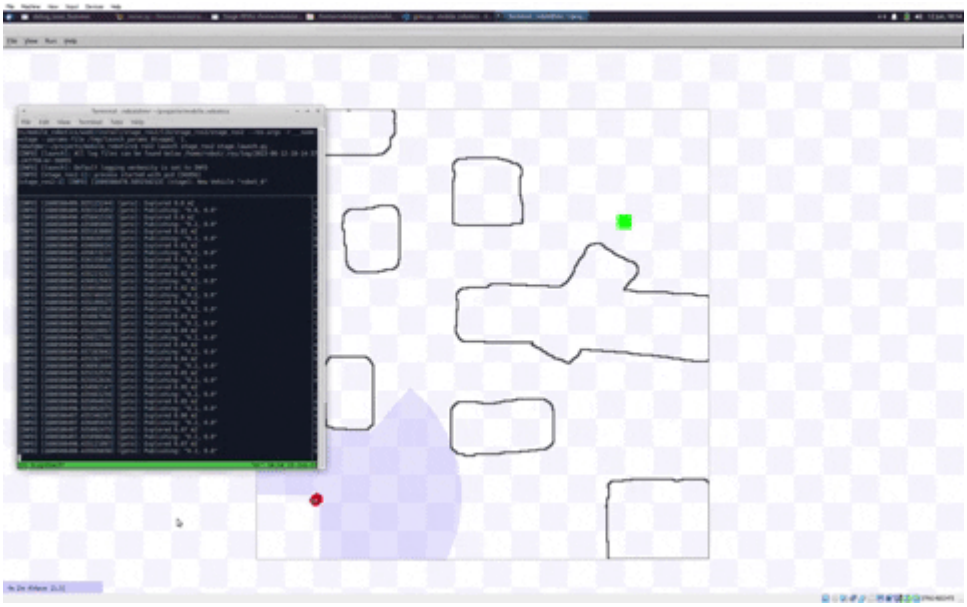
1.2 Simple, no Obstacle

To accomplish the simple goto exercise, the map was divided into grid cells of equal size. These cells all stored the cost (distance) to the cell containing the goal point. The robot navigation was executed with a simple greedy selection for the cell with the lowest distance of all the cells neighbouring the robots current position.

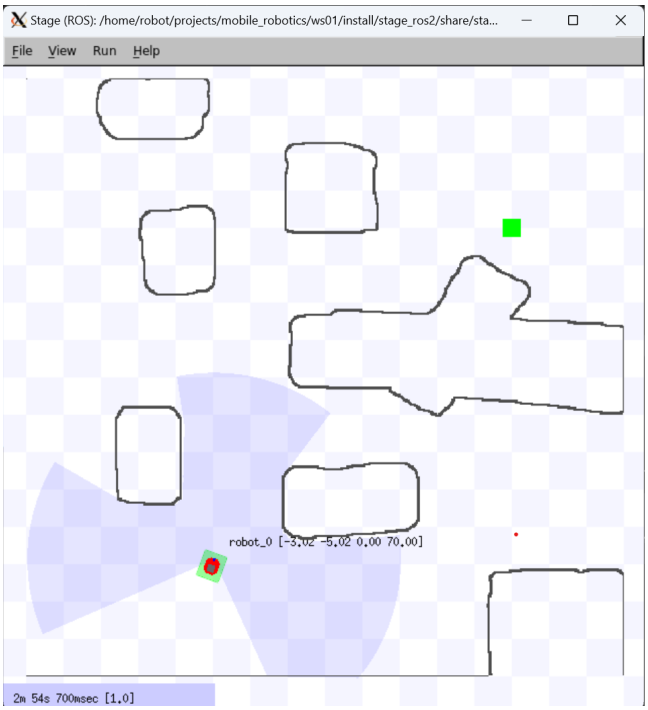
Used Command:

```
ros2 run mr_goto goto --ros-args -p mode:="plan" -p x:=-3.0 -p y:=-5.0 -p
deg:=70.0 --remap scan:=base_scan
```

Robot driving towards goal with just simple navigation:



Result with pose:



1.3 Avoid Obstacle

1.3.1 Box

Used Command: In order to accomplish this, the greedy selection from the last exercise was enhanced. The robot is discouraged from moving to close or turning towards obstacles, while it is encouraged to keep driving straight when near an obstacle. While this does not guarantee a shortest/optimal path, it works rather well.

```
ros2 run mr_goto goto --ros-args -p mode:="plan" -p x:=1.0 -p y:=-5.0 -p
deg:=70.0 --remap scan:=base_scan
```

Result:

1.3.2 Cave

The cave uses the same algorithm as the 1.3.1 assignment.

Used Command:

```
ros2 run mr_goto goto --ros-args -p mode:="plan" -p x:=-4.0 -p y:=1.0 -p
deg:=70.0 --remap scan:=base_scan
```

Result:

5. Launch File

5.1 Basic Launch

A simple launch file which is calling all the nodes.

```
ros2 launch mr_goto full.launch.py
```

5.2 Optional EKF or PF

To make EKF and PF or goto and move optional you first have to define the DeclareLaunchArguments localization and planner. Afterwards these LaunchArguments can be used in a opaque function to choose between the nodes.

```
localization:=ekf\
planner:=goto
```

5.3 Prameter File

The parameter for the launch files is declared via DeclareLaunchArgument.

```
world:=cave\
parameter_goto:=goto\
parameter_pf:=particle_filter.yaml
```

5.4 Relative Path for Parameter File

The relative Paths for the Parameter files is already called like in 5.3. The implementation is done via opaque functions where the relative path is altered to an absolute path.

6. Network (DDS-Security)

The needed commands for installing the sros2 package are:

```
sudo apt update
sudo apt install libssl-dev
colcon build --symlink-install --cmake-args -DSECURITY=ON
```

Then the keystore has to be generated:

```
ros2 security create_keystore <keystore name>
```

In this project the keystore is in the sros2 directory and called keystore.

The keys and certificates are created for each node.

```
ros2 security create_enclave <keystore name> <node name>
```

For the nodes to use the encryption the param `--enclave <node name>` is needed. Also three environment variables need to be set:

```
export ROS_SECURITY_KEYSTORE=<keystore name>
export ROS_SECURITY_ENABLE=true
export ROS_SECURITY_STRATEGY=Enforce
```

For this env_vars.sh has to be sourced.

The security message can be checked via:

```
sudo tcpdump -X -i any udp port 7400
```

7. Exploration

7.1 Calculate explored area

Every cell the robot passes through for the first time has it's area in square meters added to the total explored area, which is printed to the console every cycle.

Result:

```
[INFO] [1686587490.563000679] [goto]: Explored 0.0 m2
[INFO] [1686587490.563995610] [goto]: Publishing: "0.0, 0.0"
[INFO] [1686587491.015382078] [goto]: Explored 0.0 m2
[INFO] [1686587491.016099149] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587491.515540421] [goto]: Explored 0.01 m2
[INFO] [1686587491.517040183] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587492.015163243] [goto]: Explored 0.01 m2
[INFO] [1686587492.015985804] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587492.515239560] [goto]: Explored 0.01 m2
[INFO] [1686587492.516112311] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587493.015284048] [goto]: Explored 0.02 m2
[INFO] [1686587493.016047609] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587493.515448296] [goto]: Explored 0.03 m2
[INFO] [1686587493.516519998] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587494.015856459] [goto]: Explored 0.03 m2
[INFO] [1686587494.017199231] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587494.515606451] [goto]: Explored 0.04 m2
[INFO] [1686587494.516426173] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587495.015330759] [goto]: Explored 0.05 m2
[INFO] [1686587495.016408490] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587495.515385956] [goto]: Explored 0.05 m2
[INFO] [1686587495.516509788] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587496.015686169] [goto]: Explored 0.05 m2
[INFO] [1686587496.020134085] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587496.515691792] [goto]: Explored 0.06 m2
[INFO] [1686587496.516298683] [goto]: Publishing: "0.2, 0.0"
[INFO] [1686587497.015180769] [goto]: Explored 0.07 m2
```

7.2 Service Call

The service is implemented in the goto.py file. The command to check it is:

```
ros2 service call /explored_area std_srvs/srv/Trigger
```

Result:

```
robot@mr:~/projects/mobile_robotics$ ros2 service call /explored_area std_srvs/srv/Trigger
waiting for service to become available...
requester: making request: std_srvs.srv.Trigger_Request()

response:
std_srvs.srv.Trigger_Response(success=True, message='Explored 0.11 m2')
```

7.3 Topic

The explored are is published on the explored topic. Checking this topic while the robot is running gives the current explored area:

```
robot@mr:~/projects/mobile_robotics$ ros2 topic echo explored
data: Explored 0.39 m2
```