Bachelor Project: Interactive Mobile Manipulation: Building Affordance-Aware 3D Scene Graphs for Autonomous Robot Arm Control

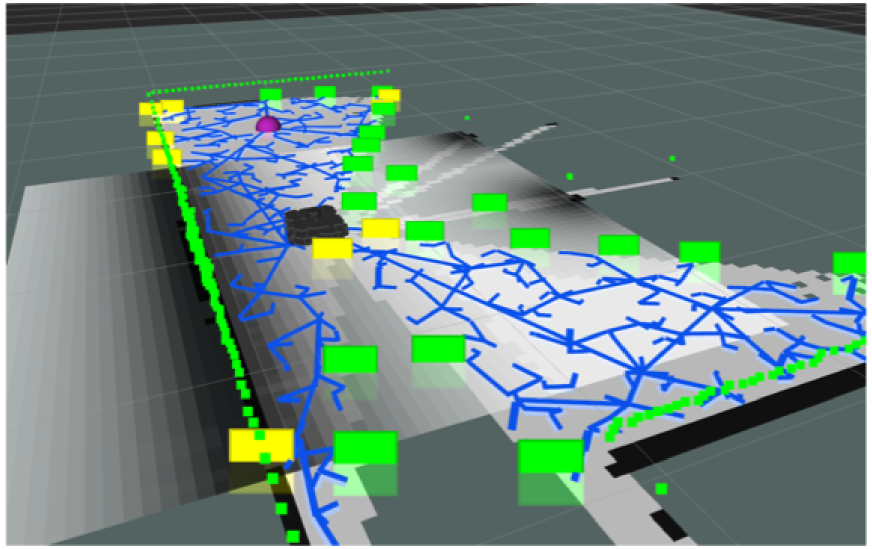
Module 3: Exploration, Navigation & Mapping

Ibrahim Sayed, Mohammed Daoudi, Ilde Tagueu

Universität Duisburg-Essen
Networked Embedded Systems Group
Faculty of Computer Science
Schützenbahn 70
D-45127 Essen, Germany

Module 3: Exploration, Navigation & Mapping

https://www.youtube.com/watch?v=aABID3RVOc8



- Hardware and software setup
  - Mobile Robot Platform (Panther UGV, TurtleBot)
  - Localization Sensors (2D and 3D LiDAR)
  - Onboard Computer (NVIDiA AGX ORIN)
  - Robot Middleware (ROS2 Yazzy)
  - Workstation
- Idea
  - Create and update local maps using SLAM algorithms
  - Plan and execute safe navigation to specified goals and waypoints
  - Implement autonomous exploration algorithms for automatic coverage of unknown areas and map expansion
  - Adapt exploration and navigation strategies based on real-time scene graph inputs to support intelligent decision-making

[1]

1

Milestone 1

Set up the robot simulation with sensors and perform SLAM using manual teleoperation.

– Show how the robot build a map in simulation with sensors and talk about SLAM approaches.

UNIVERSITÄT DUISBURG ESSEN

Open-Minded

Workspace and Build

- Create ROS 2 workspace (husarion_ws)
- Clone Husarion UGV packages
- Configure simulation environment

Build workspace using colcon

```
vcs import src < src/husarion_ugv_ros/husarion_ugv/${HUSARION_ROS_BUILD_TYPE}_deps.repos

sudo rosdep init
rosdep update --rosdistro $ROS_DISTRO
rosdep install --from-paths src -y -i

source /opt/ros/$ROS_DISTRO/setup.bash
colcon build --symlink-install --packages-up-to husarion_ugv --cmake-args -DCMAKE_BUILD_TYPE=Release -DBUILD_

source install/setup.bash
```

Launch Simulation

- Launch Husarion robot in Gazebo
- Use simulation clock (use_sim_time)
- Validate robot state and TF frames in RViz

LiDAR Integration

- 2D LiDAR Sensor
- components.yaml
- /scan
- SLAM

component.yaml:


```yaml
components:
  - type: LDR06
    parent_link: base_link
    xyz: 0.15 0.0 0.25
    rpy: 0.0 0.0 0.0
    frequency: 10.0
    horizontal_samples: 720
    horizontal_fov: 3.14159
    range_min: 0.12
    range_max: 12.0
    noise_stddev: 0.01
    topic: /scan
    frame_id: laser_front_link
```

SLAM Toolbox

SLAM Toolbox (Online Mapping)

- Installed `slam_toolbox` package
- Online async SLAM mode
- Uses `/scan` + odometry to build map
- Map visualized in RViz (Occupancy Grid)

Issue: Wrong scan topic subscription at first
Fix: Use correct `/scan` topic → map updates correctly

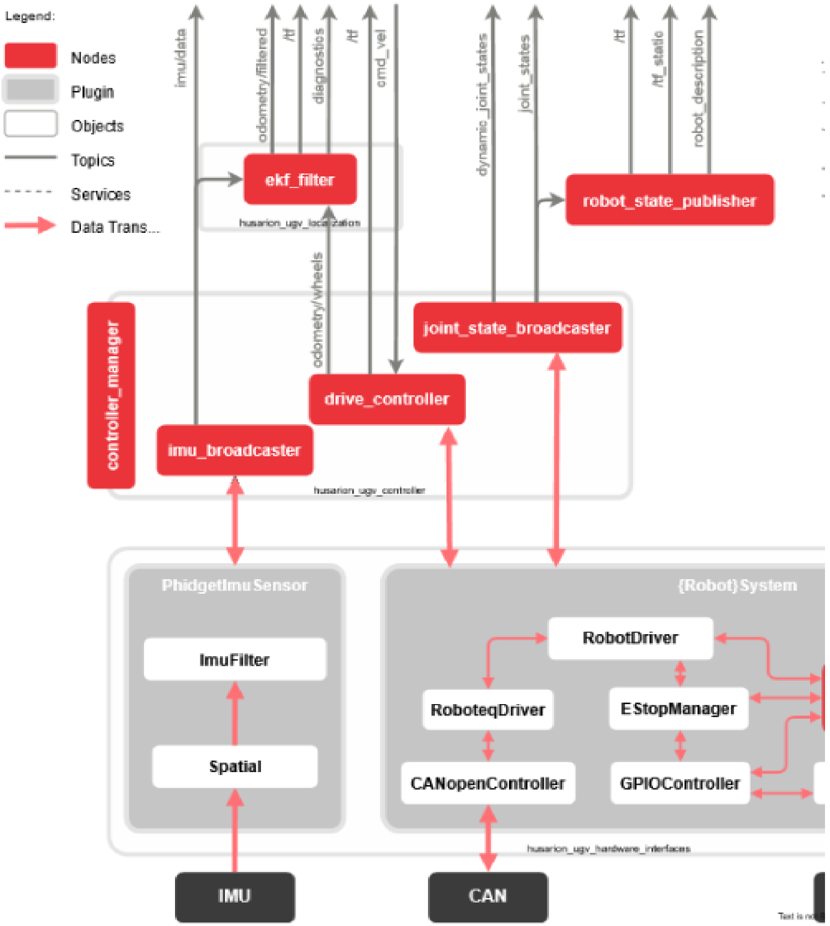
2

Milestone 2

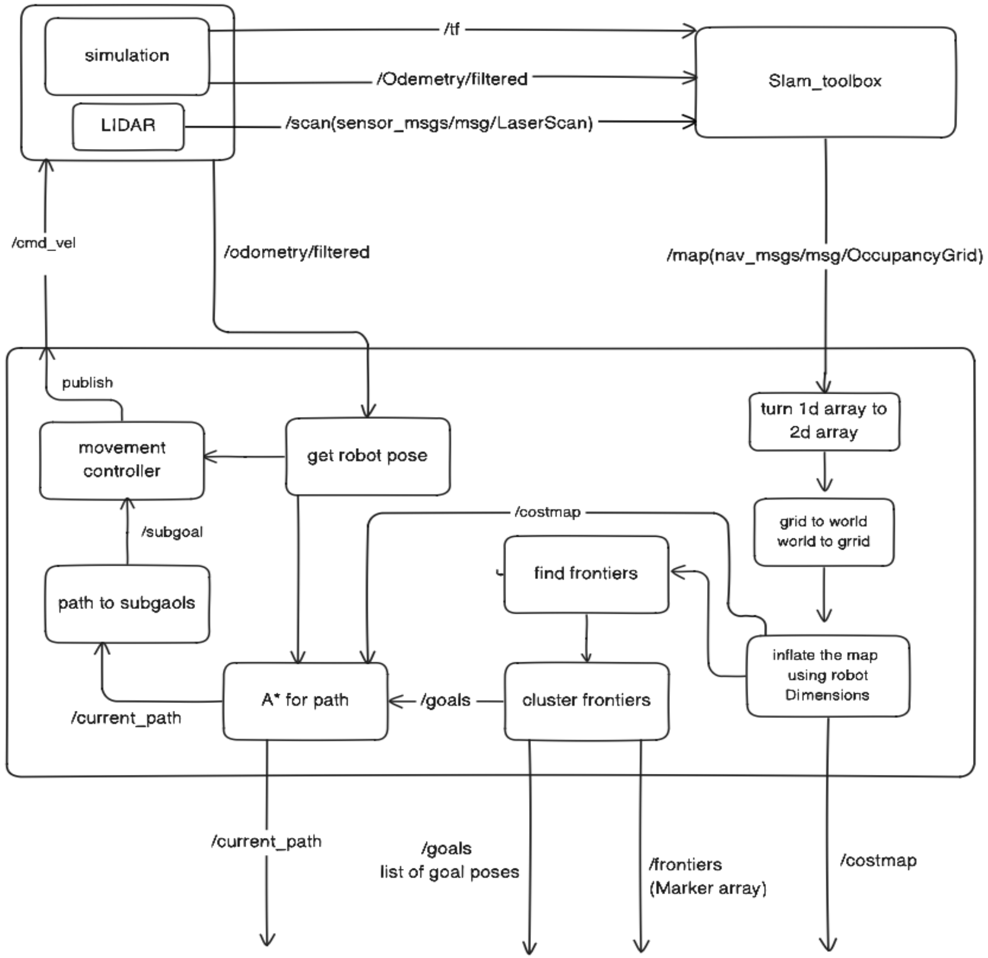
Implement autonomous exploration capabilities.

– Show the robot exploring autonomously in an unknow environment while simultaneously creating a map.

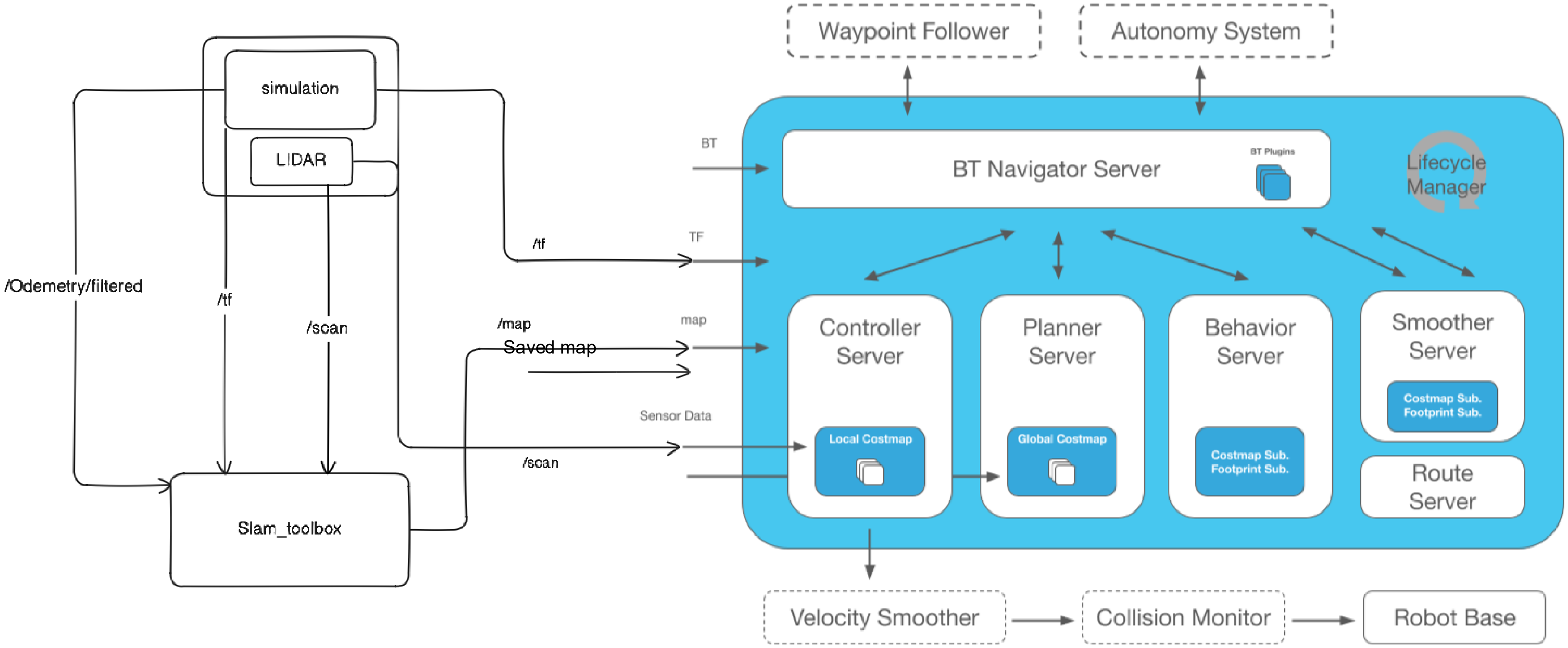
Custom navigation Data flow



[2]



Nav2



simulation
LIDAR
/Odemetry/filtered
/tf
/scan
Slam_toolbox
/tf
/map
Saved map
/scan
BT
TF
map
Sensor Data
Waypoint Follower
Autonomy System
BT Navigator Server
BT Plugins
Lifecycle Manager
Controller Server
Local Costmap
Planner Server
Global Costmap
Behavior Server
Costmap Sub.
Footprint Sub.
Smoother Server
Costmap Sub.
Footprint Sub.
Route Server
Velocity Smoother
Collision Monitor
Robot Base

[3]

Nav2 params file

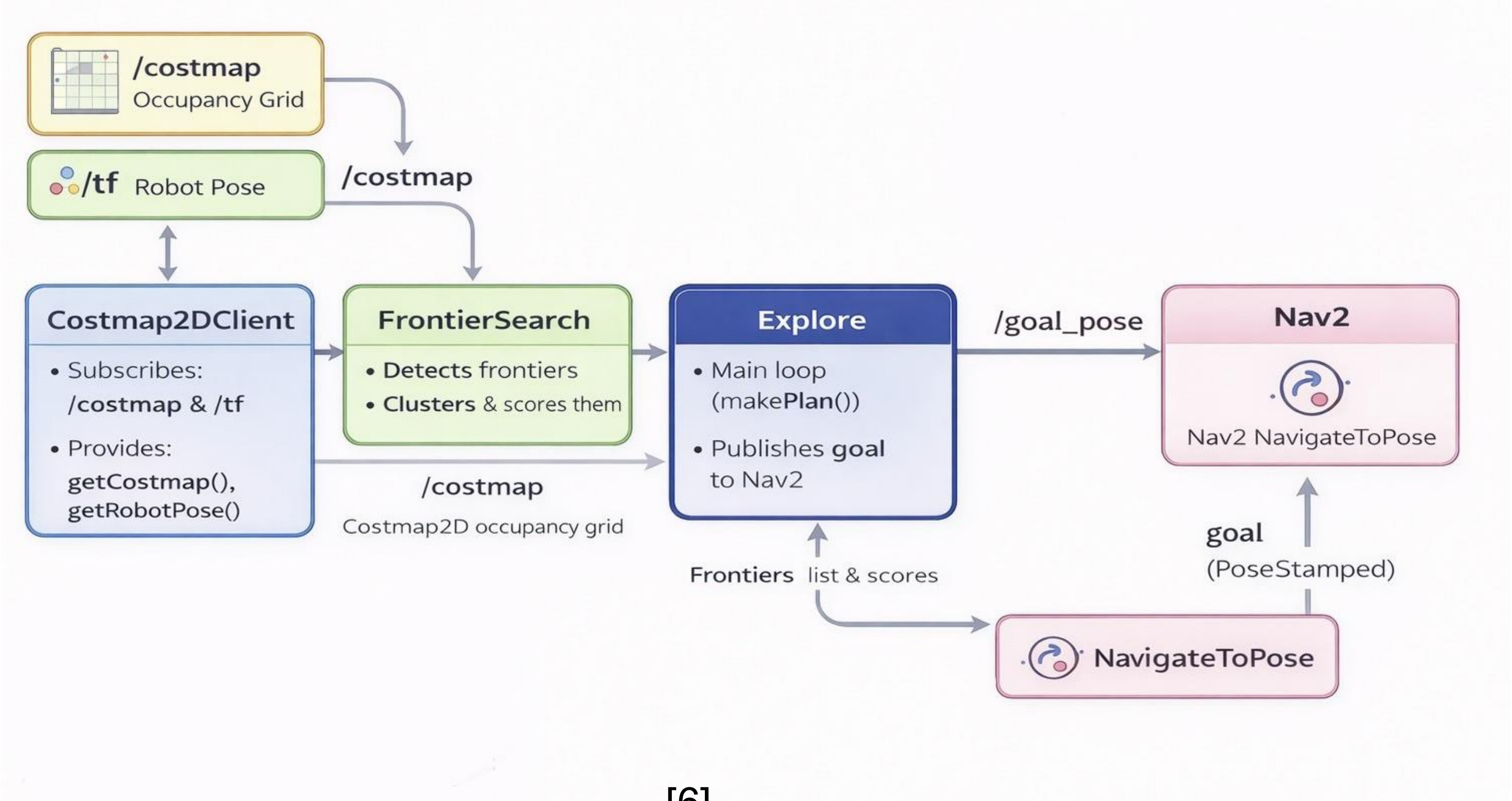
```yaml
velocity_smoother:
  odom_topic: "/odometry/filtered"


bt_navigator:
  ros__parameters:
    odom_topic: /odometry/filtered
```

```yaml
global_costmap:
  global_costmap:
    ros__parameters:
      robot_base_frame: base_link
      robot_radius: 0.6 #0.22
      resolution: 0.05

      inflation_layer:
        cost_scaling_factor: 10.0 #3.0
        inflation_radius: 0.5 #0.7

local_costmap:
  local_costmap:
    ros__parameters:
      global_frame: odom
      robot_base_frame: base_link
      resolution: 0.05
      robot_radius: 0.55 #0.22

      inflation_layer:
        plugin: "nav2_costmap_2d::InflationLayer"
        cost_scaling_factor: 8.0 #3.0
        inflation_radius: 0.5
```

```yaml
bt_navigator:
  ros__parameters:
    enable_stamped_cmd_vel: true

controller_server:
  ros__parameters:
    enable_stamped_cmd_vel: true

behavior_server:
  ros__parameters:
    enable_stamped_cmd_vel: true


velocity_smoother:
  ros__parameters:
    enable_stamped_cmd_vel: true


collision_monitor:
  ros__parameters:
    enable_stamped_cmd_vel: true


docking_server:
  ros__parameters:
    enable_stamped_cmd_vel: true
```

- Change odom topic
- Adjust the infaltion to match the robot
- Change publishing msg type for /cmd_vel to be twistStamped

Autonomous exploration: explore_lite



[6]

Reading Data : Costmap2DClient

- **Role:** Provides real-time data to the node
- **Functions:**
  - Subscribes to:
    - `/costmap` (OccupancyGrid from Nav2)
    - `/costmap_updates`
  - Uses TF2 to get the transform from `map` → `base_link`

Provides:

- `getCostmap()` → used by `FrontierSearch`
- `getRobotPose()` → used in `makePlan()` to know where the robot is

FrontierSearch

- Takes costmap and robot pose
  - Detects frontier cells:
    - Unknown (-1) cells next to free space (0)
  - Clusters frontier cells with **BFS**
  - Computes cost for each frontier:
    - Cost = `potential_scale * distance - gain_scale * size`
  - Returns sorted list of frontiers

Explore Node

- Periodically runs `makePlan()` (via ROS2 timer)
- In each cycle:
  - Get robot position from `Costmap2DClient`
  - Run `FrontierSearch.searchFrom()`
  - Filter out blacklisted goals
  - Select best frontier (lowest cost)
  - Convert centroid to PoseStamped
  - Send goal to Nav2
  - **Handles Feedback:**
- If goal is aborted: it is blacklisted
- If successful: next cycle continues

3

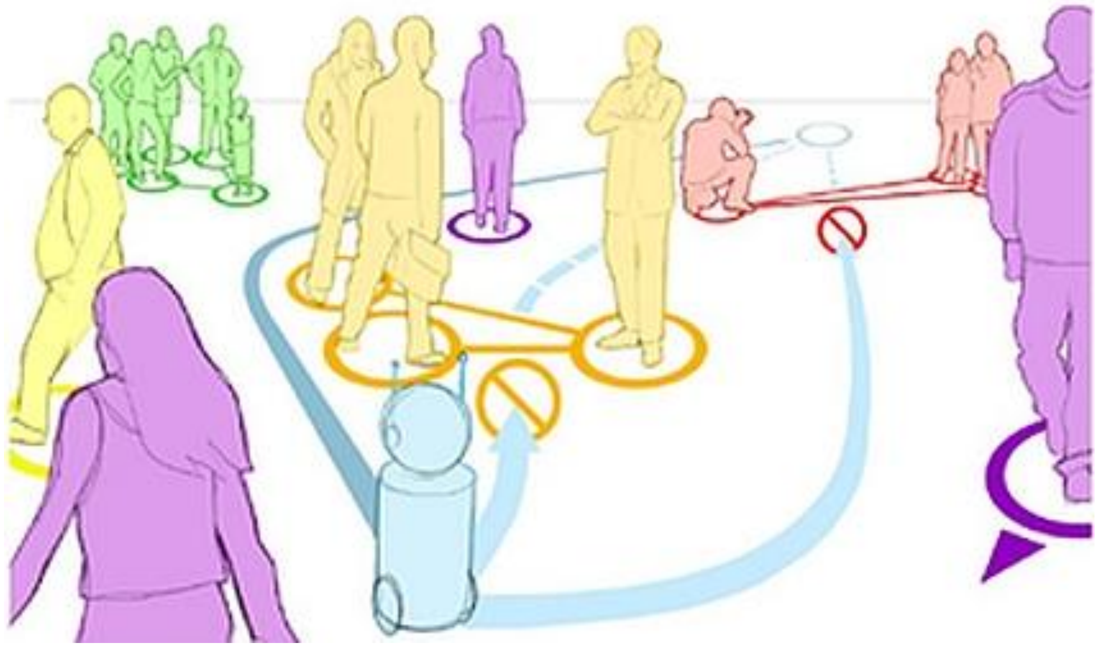
Milestone 3

Develop and integrate social navigation behavior.

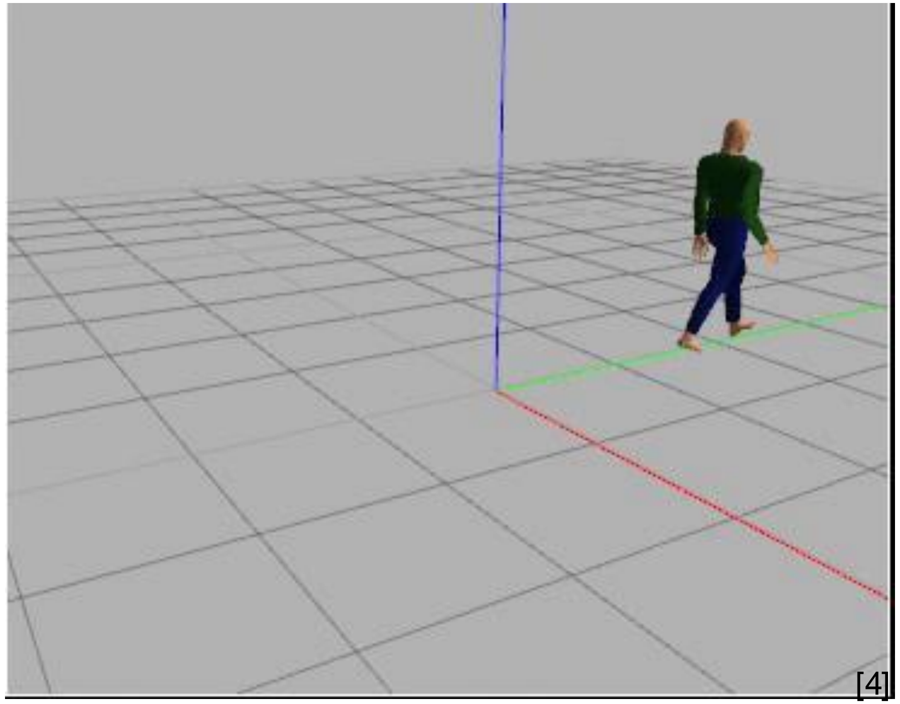
– Show the robot navigation from one position to another without crashing static and dynamic objects (persons)

Dynamic Objects in Gazebo

In Gazebo we added 4 persons that moves in a loop with square pattern



[5]



[4]

Nav2 and social robotics

- To avoid dynamic objects the robot needs to understand objects and predict their trajectory (Social)

  but Nav2 **does NOT "understand objects"**
  so how did it avoids dynamic objects??

  Answer: Continuous sensor updates → costmap update → local planner reacts

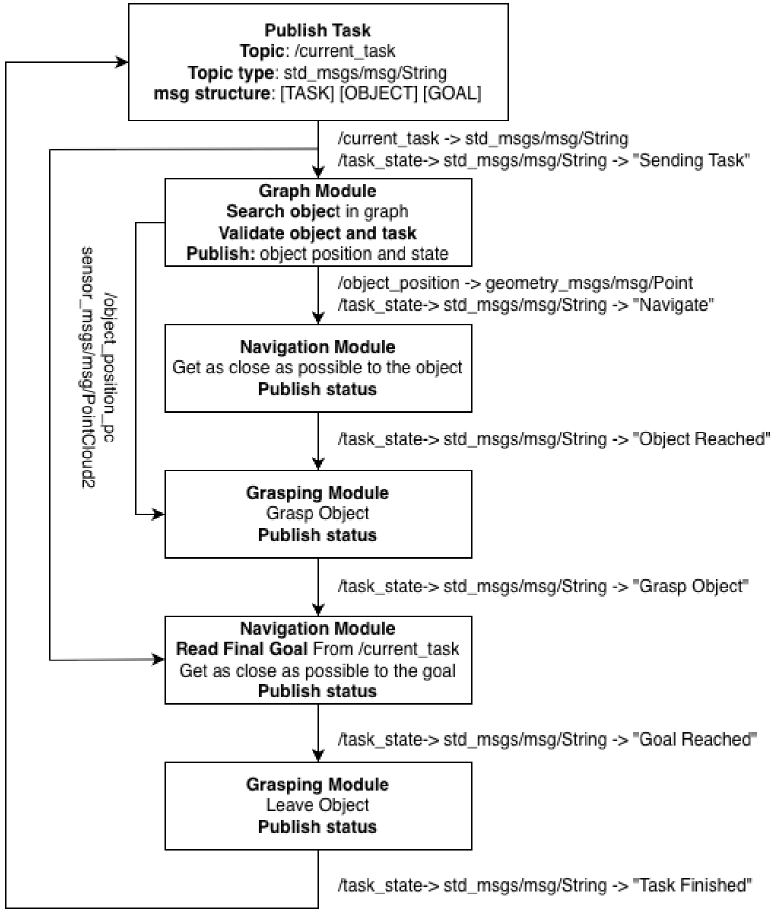
4

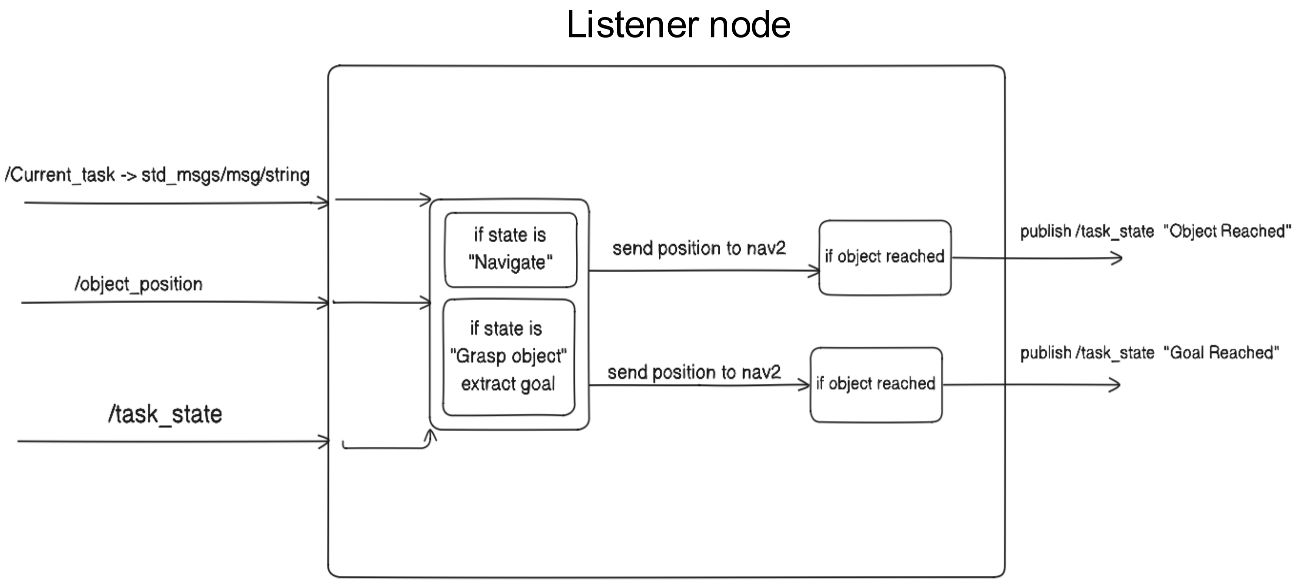
Milestone 4

Achieve full integration with other system modules.

– Show the complete system running with the other modules

Integration Diagram





Questions?

If you really want to

References

[1] https://lehre.moodle.uni-due.de/pluginfile.php/924226/mod_resource/content/1/Session%202.1%20-%20Exploration%2C%20Navigation%20and%20Mapping.pdf

[2] https://husarion.com/manuals/panther/ros2-api/

[3] https://docs.nav2.org/

[4] https://classic.gazebosim.org/tutorials?tut=actor

[5] http://www.spencer.eu/project.html

[6] https://wiki.ros.org/explore_lite