FOSDEM

I'm giving a talk:

WHALES USE LIGHTHOUSES TOO: OPEN SOURCE POSITIONING FOR OPEN SOURCE

2nd February, 2025

Robotics and Simulation devroom, UB2.147

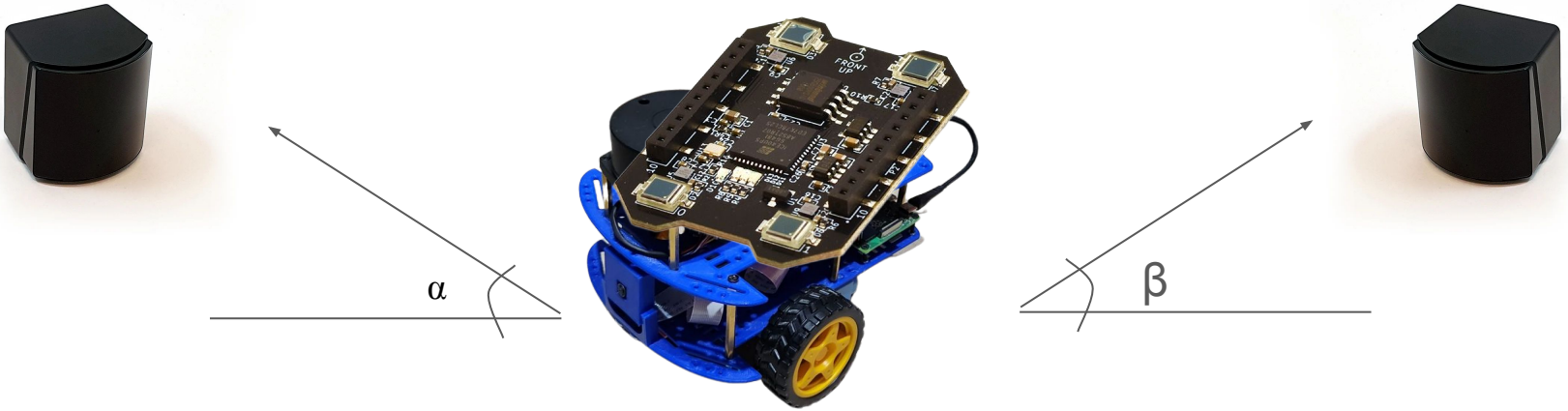
fosdem.org/2025/schedule/track/robotics/

EKUMEN

BELUGA

Gerardo Puga, Gonzalo de Pedro, Juan Manuel Carosella, Michel Hidalgo

Lighthouse Positioning system



Lighthouse positioning deck: timing info from the base stations scans.

Lighthouse_ros: Computes azimuth and elevation angles to the base stations.

Enter Beluga

Open source Monte Carlo Localization (MCL) toolkit, focused on code quality and performance.

- AMCL drop-in replacement node
  - ROS1 and ROS2 support
- Extensible, ROS independent, MCL library.

Beluga as a library

```
1.    auto update(state_type control_action, measurement_type measurement) -> estimation_type {
2.      particles_ |= beluga::actions:: propagate(motion_model_ (window << std:: move(action))) |
3.                    beluga::actions:: reweight(sensor_model_ (std::move(measurement)))       |
4.                    beluga::actions:: normalize();
5.
6.      particles_ |= beluga::views::sample                    |
7.                    beluga::views:: take_while_kld(params) |
8.                    beluga::actions::assign;
9.
10.     return beluga::estimate(beluga::views:: states(particles_),
    beluga::views:: weights(particles_));
11.   }
```

Two ways to adapt Beluga to our needs:

- Write a new `sensor_model`, and use Beluga AMCL algorithm.
- Make a new `update` method, using the lower level APIs.
  - Maybe if we want to fuse lidar and lighthouse

Beluga as a library

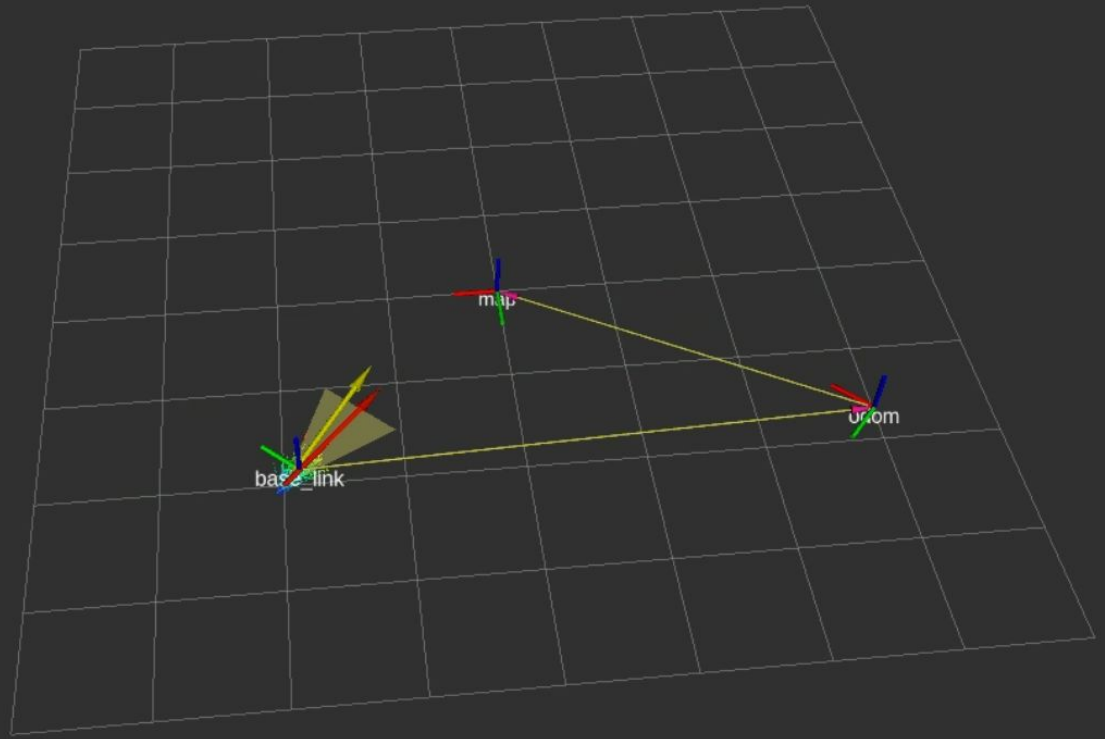
- Write `LightHouseSensorModel2D`

```
1.  LightHouseSensorModel2D (param_type params, LighthouseStationMapType landmark_map) {}
2.
3.  /// Returns a state weighting function conditioned on lighthouse deck
4.  /// measurements
5.  auto operator()(measurement_type &&detections)  const {
6.   return [this, detections = std::move(detections)](const state_type &state) -> weight_type
7.   {
8.     (...)
9.     return direction_error_prob;
10.  };
11.  }
```

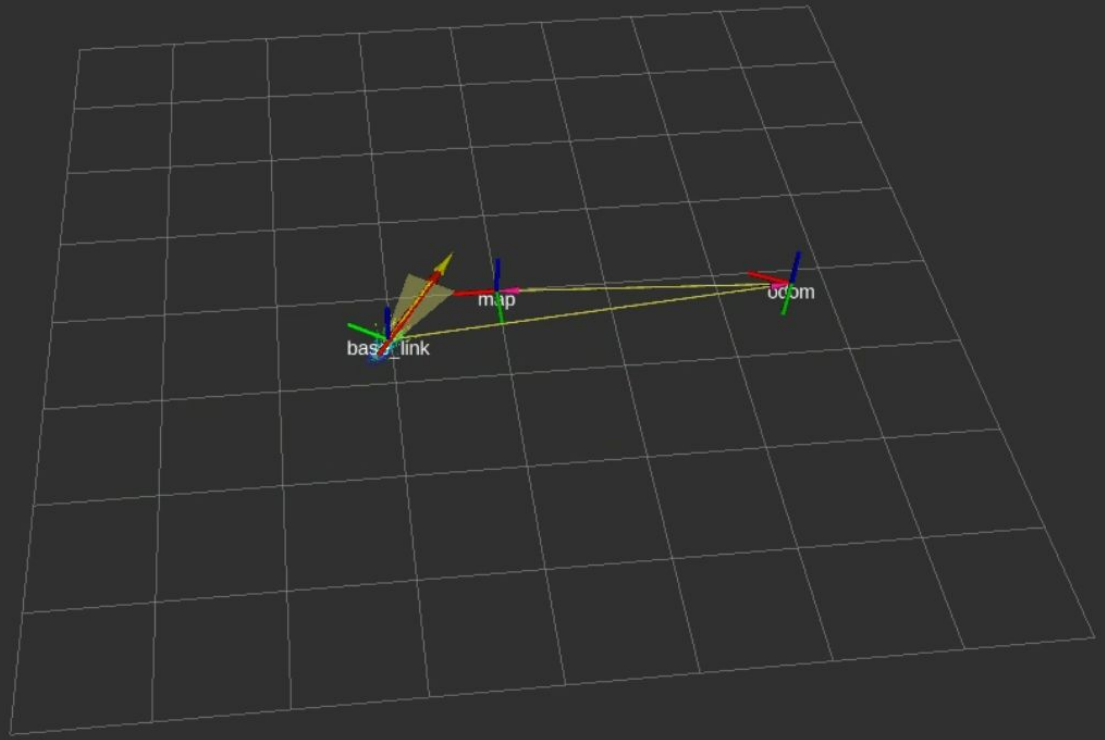
- Write lighthouse_amcl ROS node wrapping Beluga AMCL:
  - Subscribes to the sensor messages
  - Runs MCL particle filter using the lighthouse sensor model

```
beluga::Amcl(MotionModel motion_model, SensorModel sensor_model,  const AmclParams& params)
```

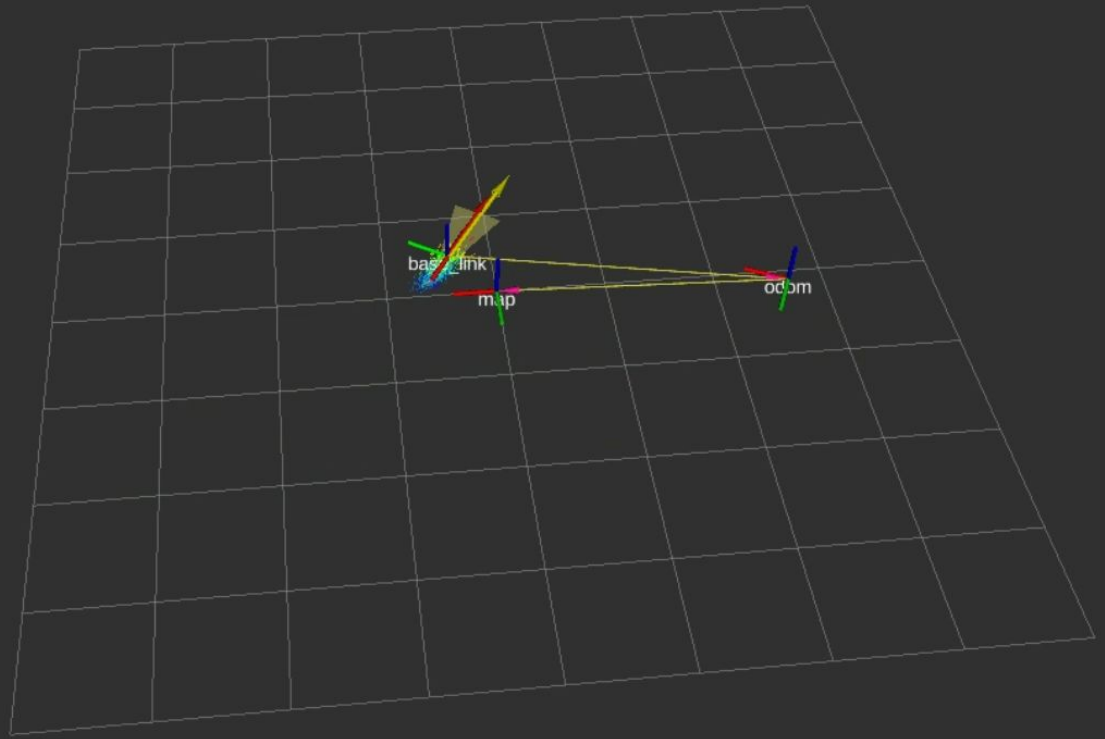
Beluga updating robot pose using simulated lighthouse base station positions



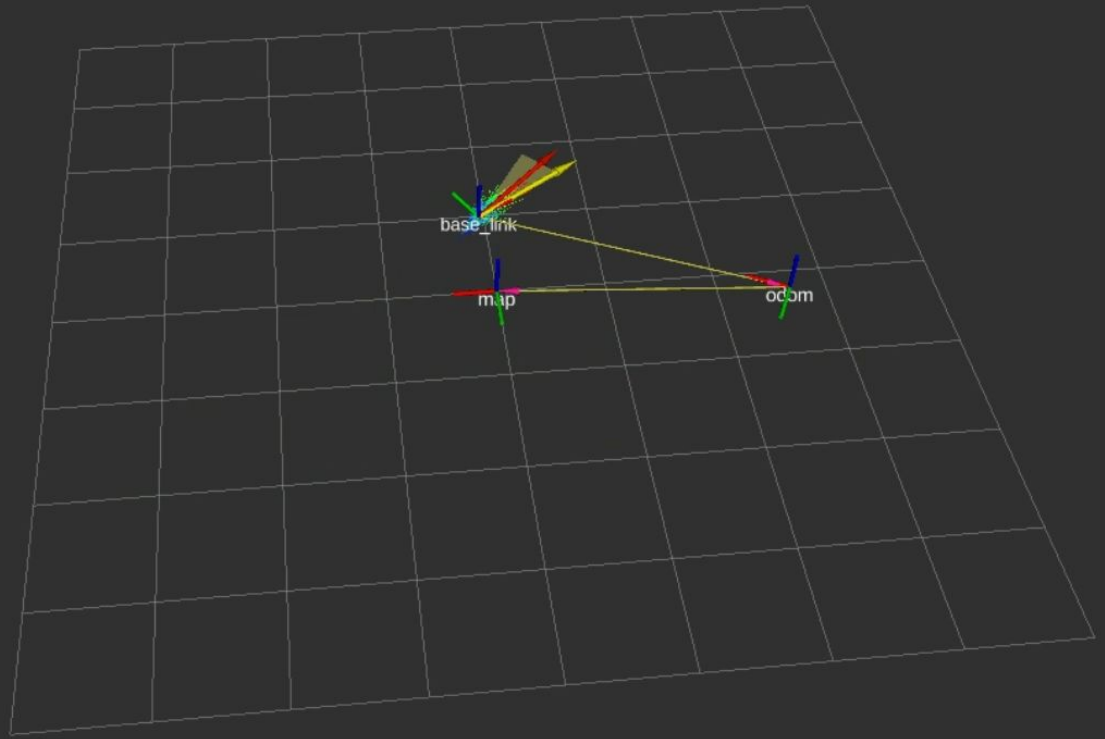
Beluga updating robot pose using simulated lighthouse base station positions



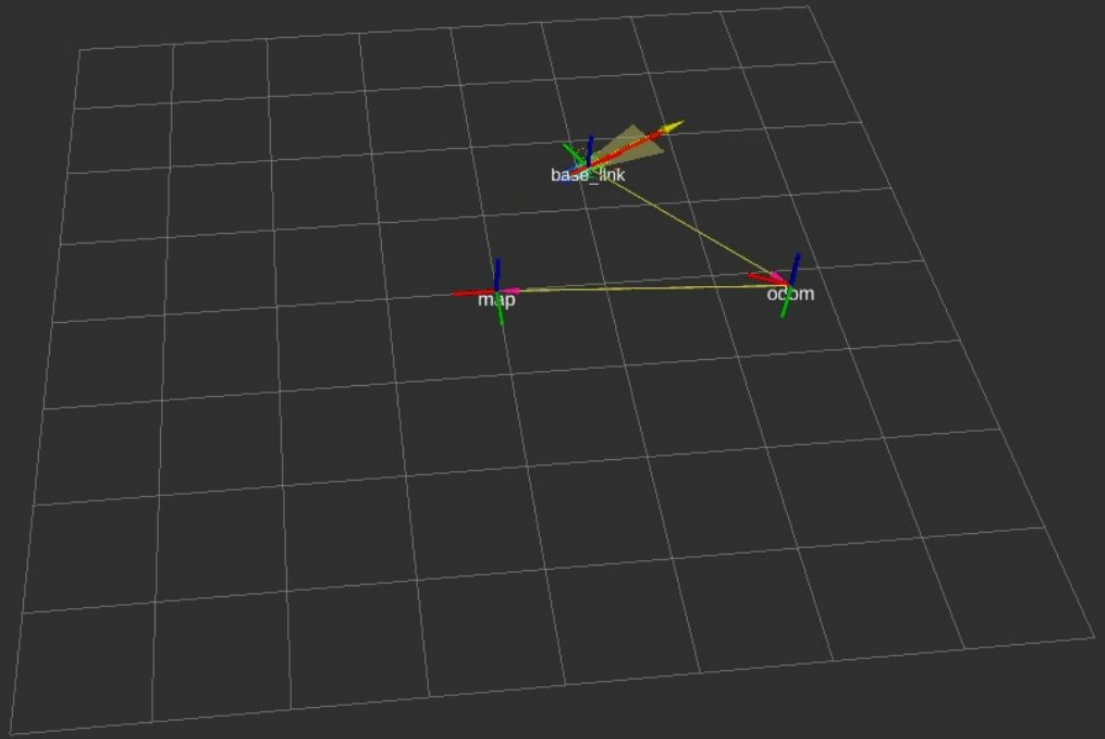
Beluga updating robot pose using simulated lighthouse base station positions



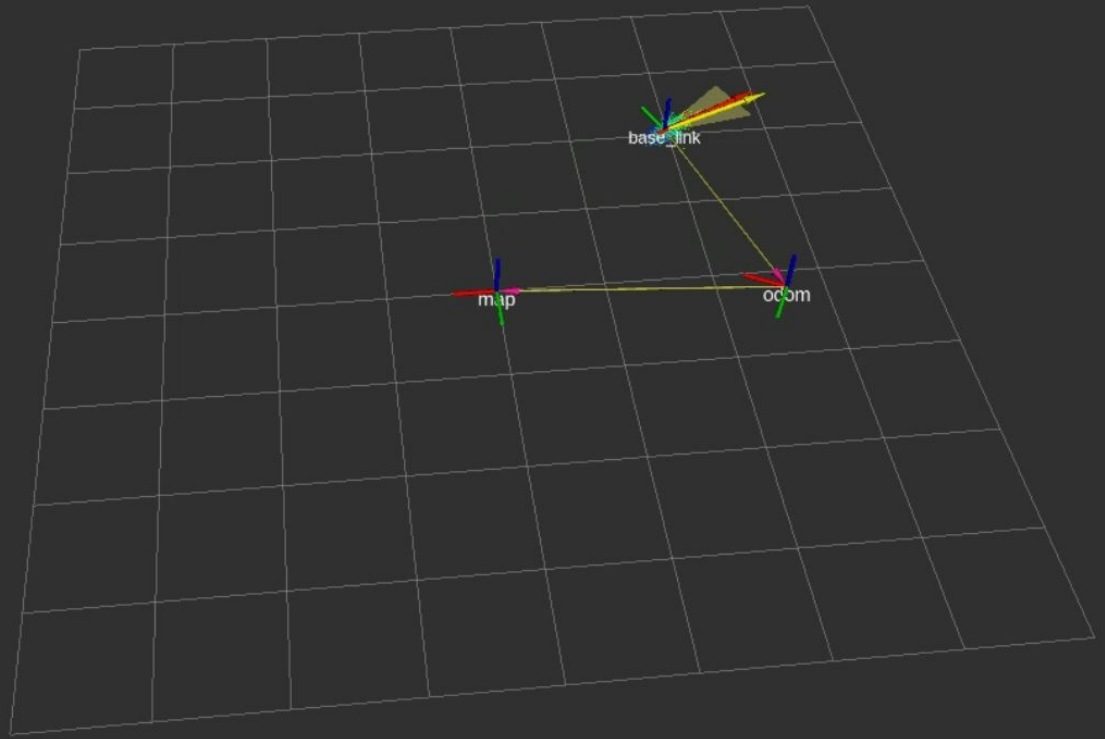
Beluga updating robot pose using simulated lighthouse base station positions



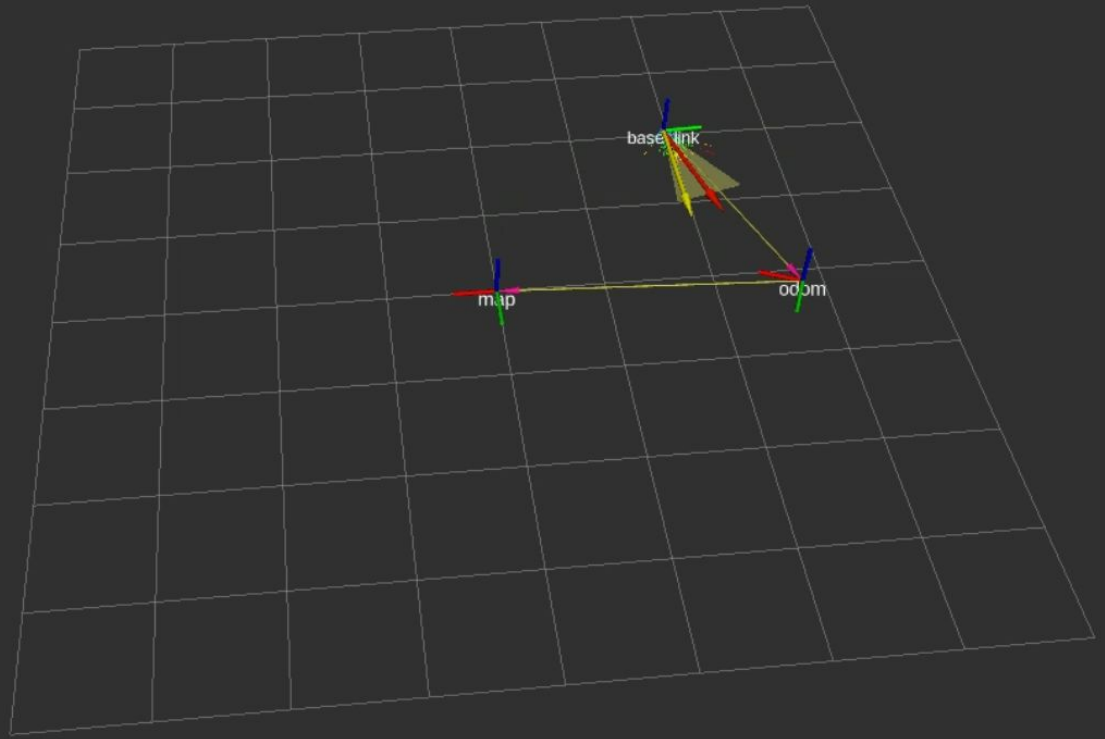
Beluga updating robot pose using simulated lighthouse base station positions



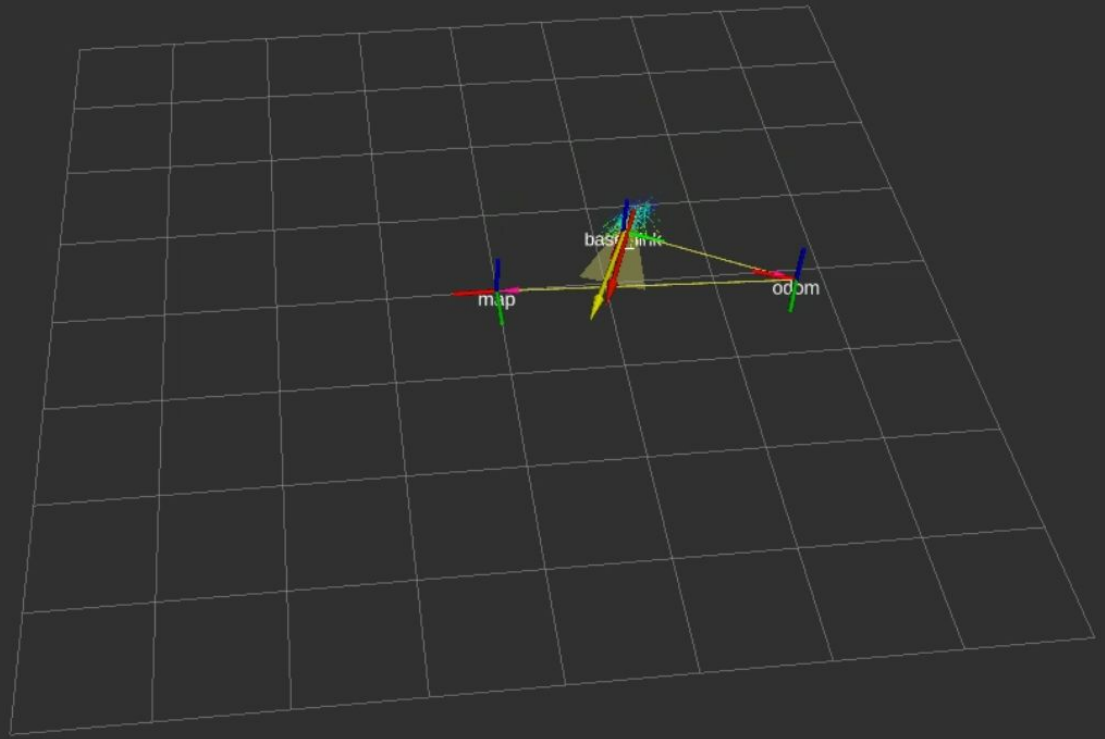
Beluga updating robot pose using simulated lighthouse base station positions



Beluga updating robot pose using simulated lighthouse base station positions



Beluga updating robot pose using simulated lighthouse base station positions



EKUMEN

Thank you!