Instituto Tecnológico de Aeronáutica

LMI

SOLVING THE SIMULTANEOUS LOCALIZATION AND 3D MAPPING PROBLEM IN MOBILE ROBOTICS USING MULTI-LEVEL PARAMETERIZED REPRESENTATIONS

Leonardo Mariga

Supervisor: Prof. Dr. Cairo L. Nascimento Jr.

Summary

- Introduction
  - Motivation
  - Problem statement
  - Objectives
- Theoretical background
  - SLAM overview
  - EKF SLAM
- Proposed solution
  - EKF framework
  - Feature extraction and update
  - Map representation
  - Integration
- Simulated results
- Experimental results
- Conclusion and future works

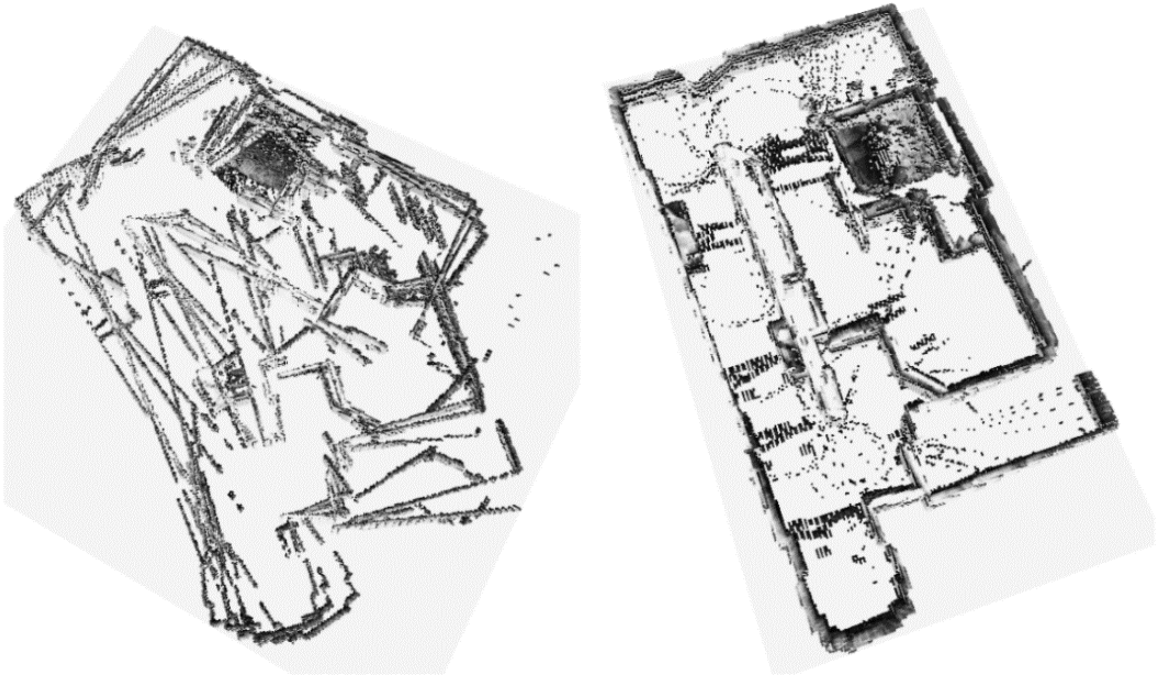
Motivation

- Use of mobile robots for 3D mapping
  - Military missions
  - Civil engineering
  - Industries

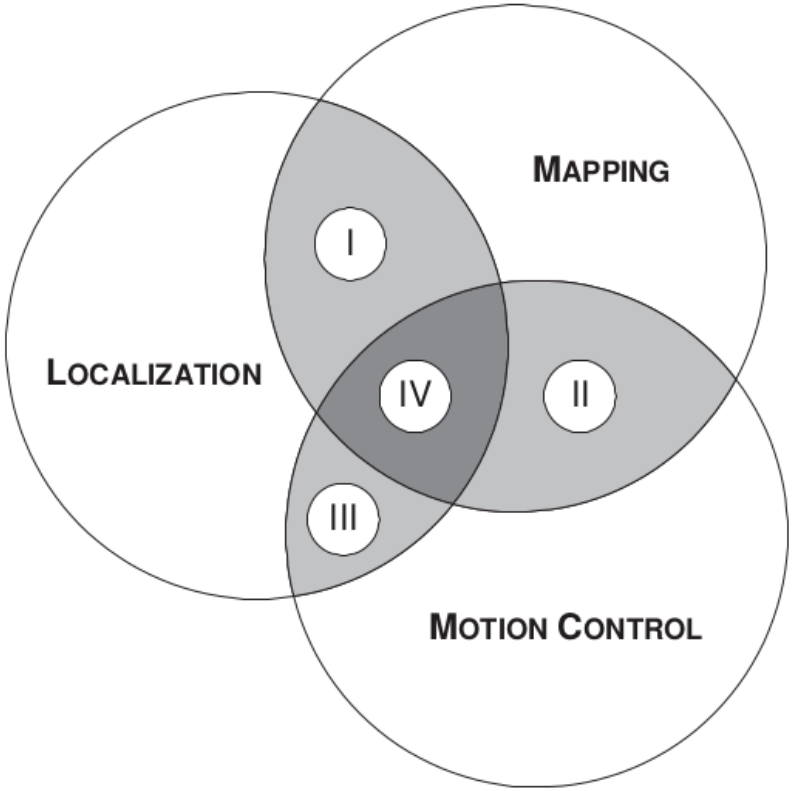
Boston Dynamics

https://youtu.be/05Zr-WQuHUU?t=18

- SLAM (Simultaneous Localization and Mapping)

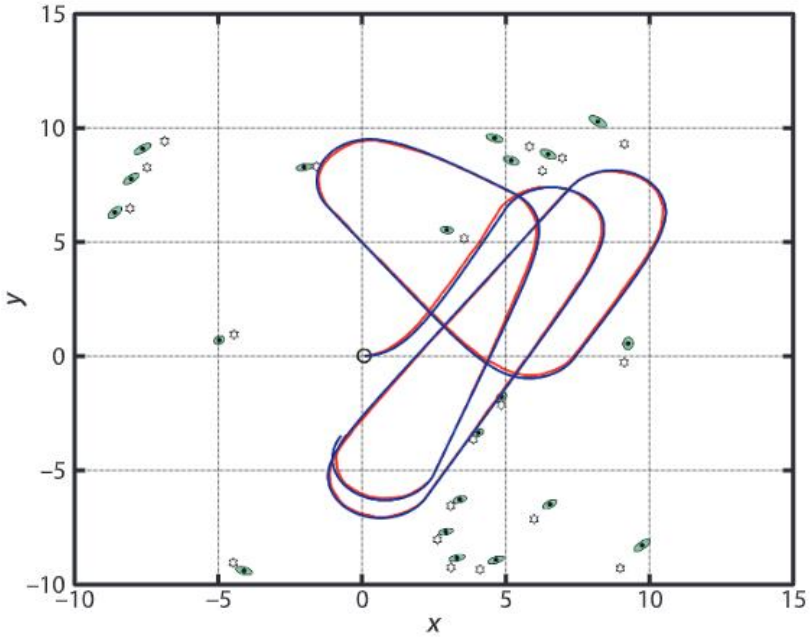


Eck (2013)

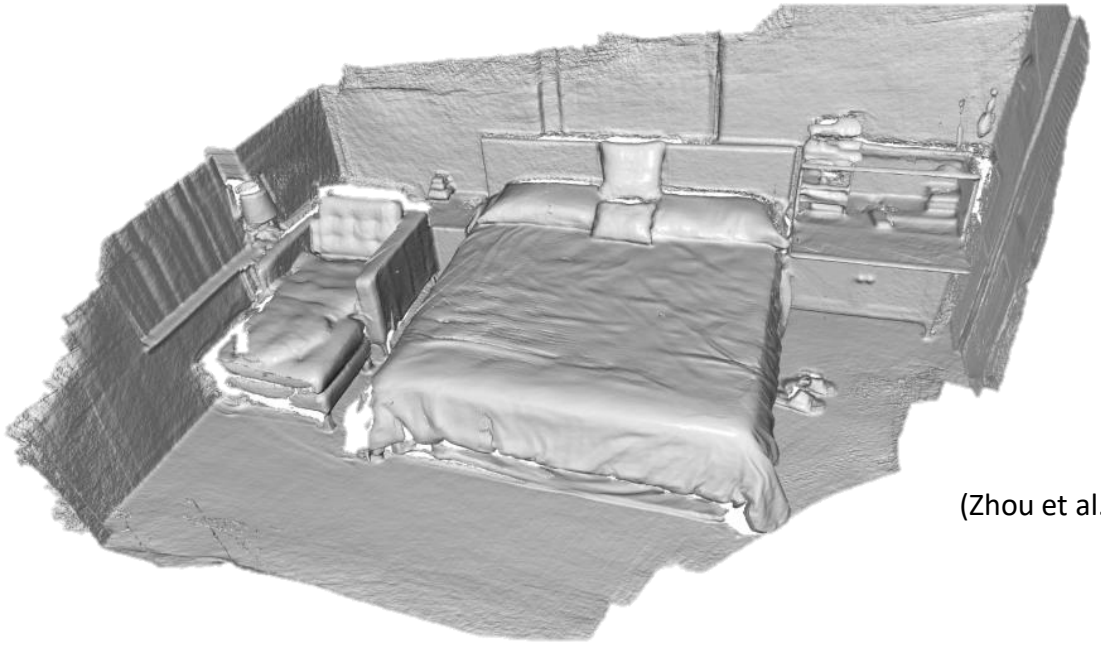


(I) SLAM, (II) Classic exploration,(III) Active localization, (IV) Integrated exploration (MAKARENKO et al., 2002).

- Classical SLAM solution (EKF-SLAM)
- Mapping representations



(CORKE, 2013)

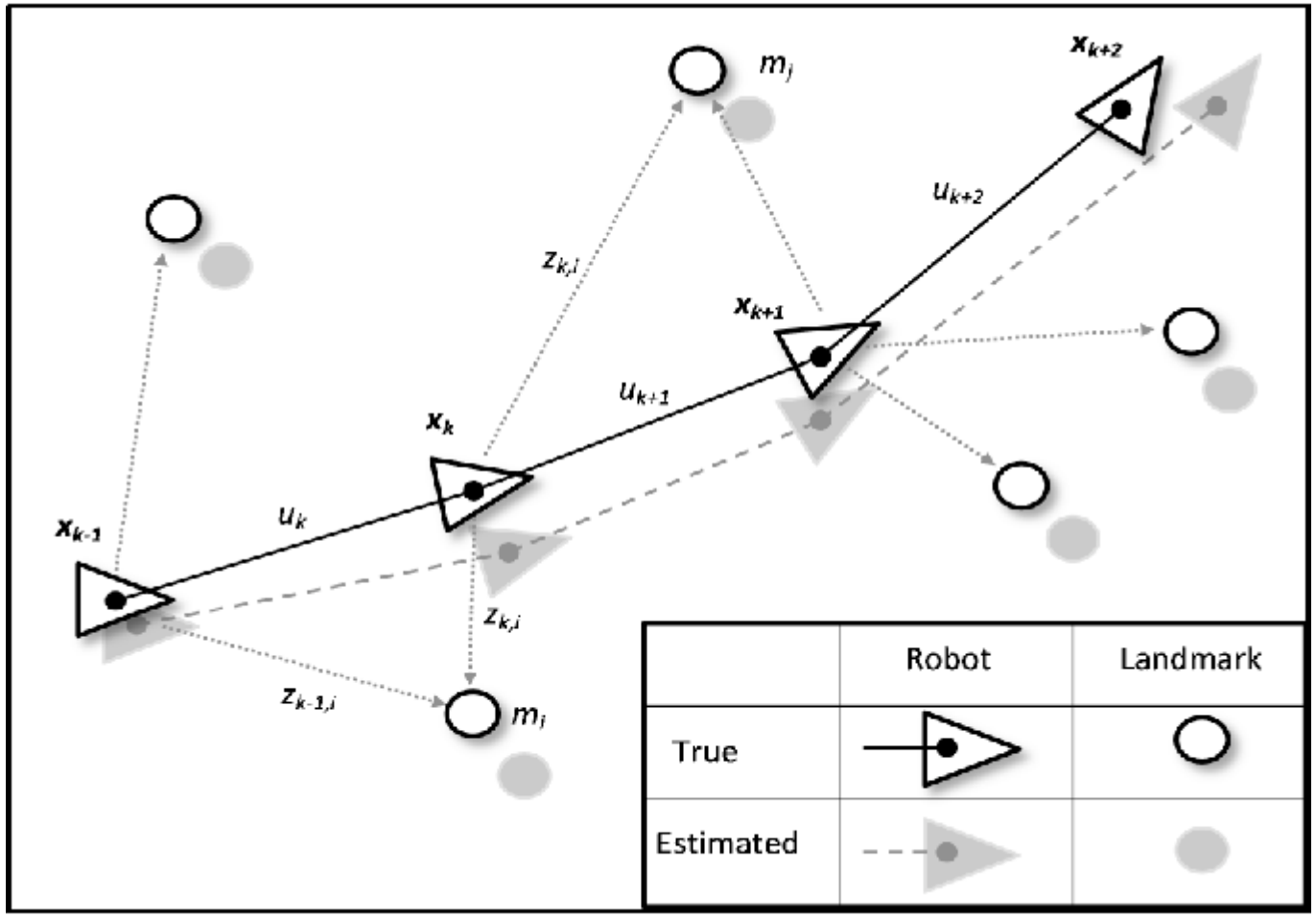


(Zhou et al. , 2013)

Objectives

- Expand the classical EKF-SLAM to a 3D indoor environment using a mobile robot.
  - Simulation
  - Experimental results
  - Compare different 3D mapping representations

SLAM



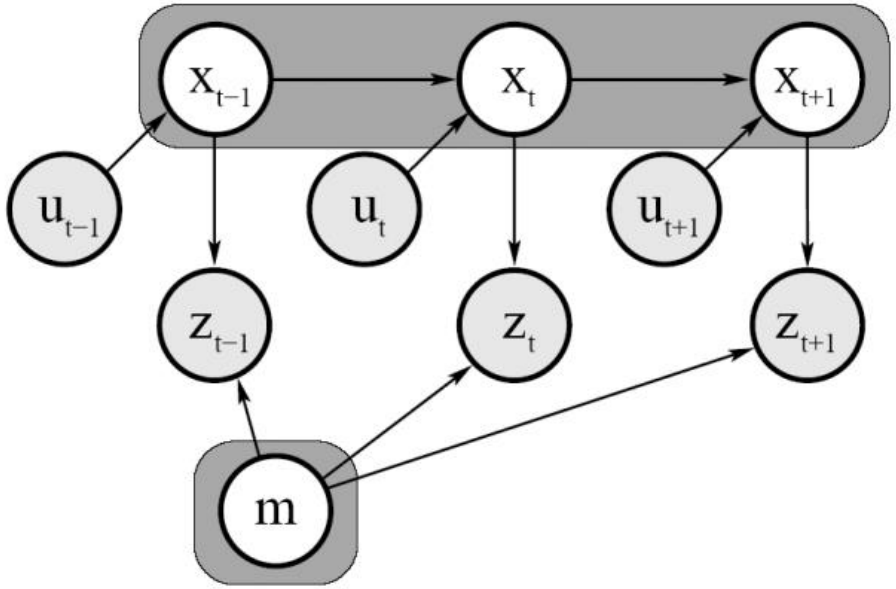
A. B. Saman, A. Lotfy.(2016)

x: robot's pose
m: map

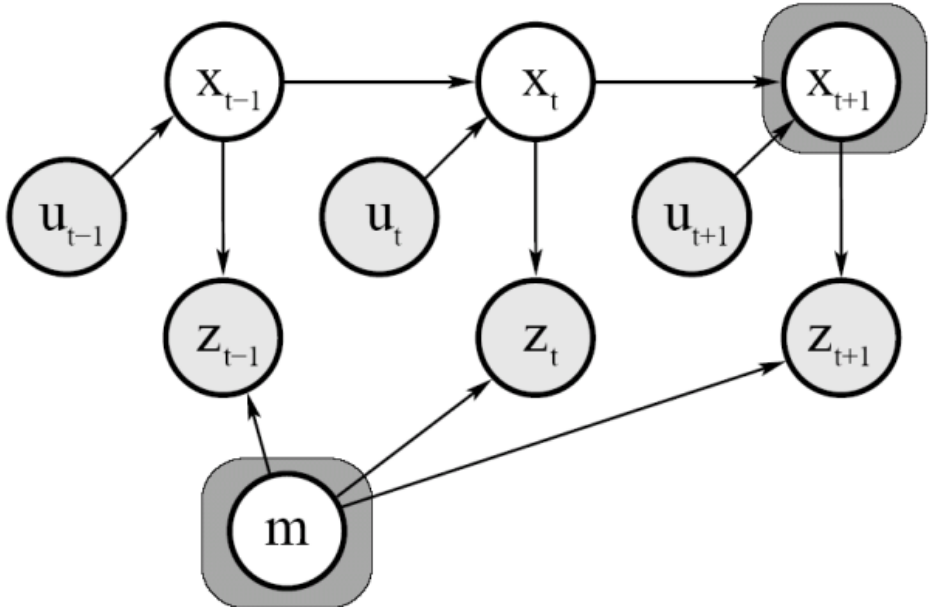
u: input
z: map measurement

- Static environment

Full SLAM



Online SLAM



(THRUN et al., 2005)

Extended Kalman Filter framework:

System model:

$$x_{k+1} = f(x_k, u_k, v_k)$$

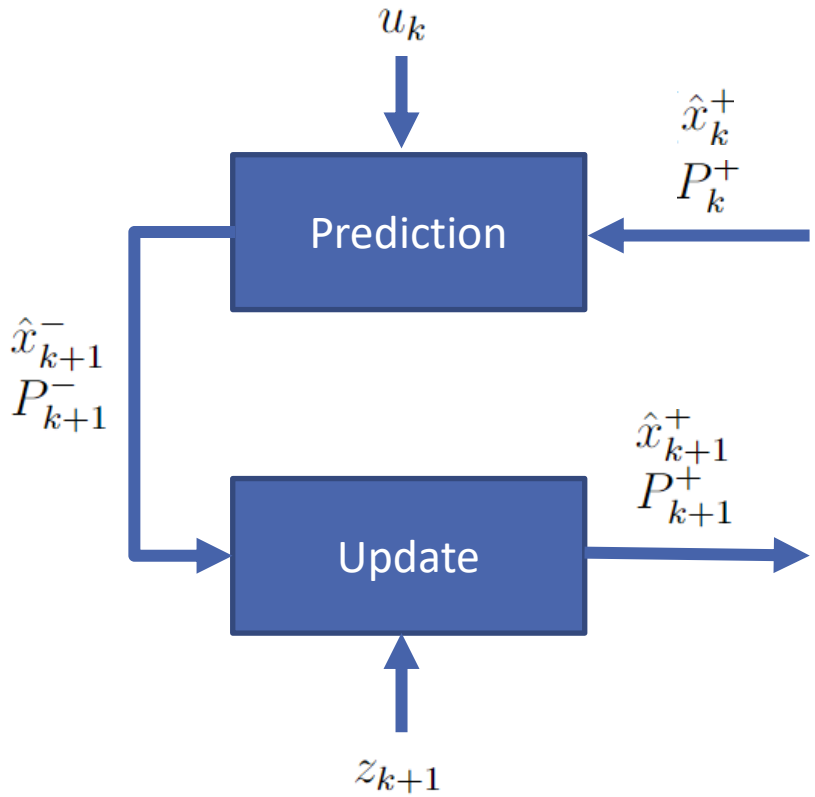
$$z_k = h(x_k, w_k)$$

Prediction Step

$$\hat{x}_{k+1}^{-} = f(\hat{x}_k^{+}, u_k)$$
$$P_{k+1}^{-} = F_x\, P_k^{+}\, F_x^T + F_v\, V_k\, F_v^T$$

Update Step

$$\nu_{k+1} = z_{k+1} - h(\hat{x}_{k+1}^{-})$$
$$L_{k+1} = P_{k+1}^{-}\, H_x^T \left(H_x\, P_{k+1}^{-}\, H_x^T + H_w\, W_{k+1}\, H_w^T\right)^{-1}$$
$$\hat{x}_{k+1}^{+} = \hat{x}_{k+1}^{-} + L_{k+1}\, \nu_{k+1}$$
$$P_{k+1}^{+} = P_{k+1}^{-} - L_{k+1}\, H_x\, P_{k+1}^{-}$$



$$F_x = \frac{\partial f(x,u,v)}{\partial x} \qquad F_v = \frac{\partial f(x,u,v)}{\partial v}$$

$$H_x = \frac{\partial h(x,w)}{\partial x} \qquad H_w = \frac{\partial h(x,w)}{\partial w}$$

Extended Kalman Filter framework in a SLAM problem:

State vector definition:

$$\hat{x} = [\underbrace{x_v,\ y_v,\ \theta_v}_{X_v},\ \underbrace{x_1,\ y_1,\ z_1}_{m_1},\ \underbrace{x_2,\ y_2,\ z_2}_{m_2},\ \ldots,\ \underbrace{x_M,\ y_M,\ z_M}_{m_M}]^T$$

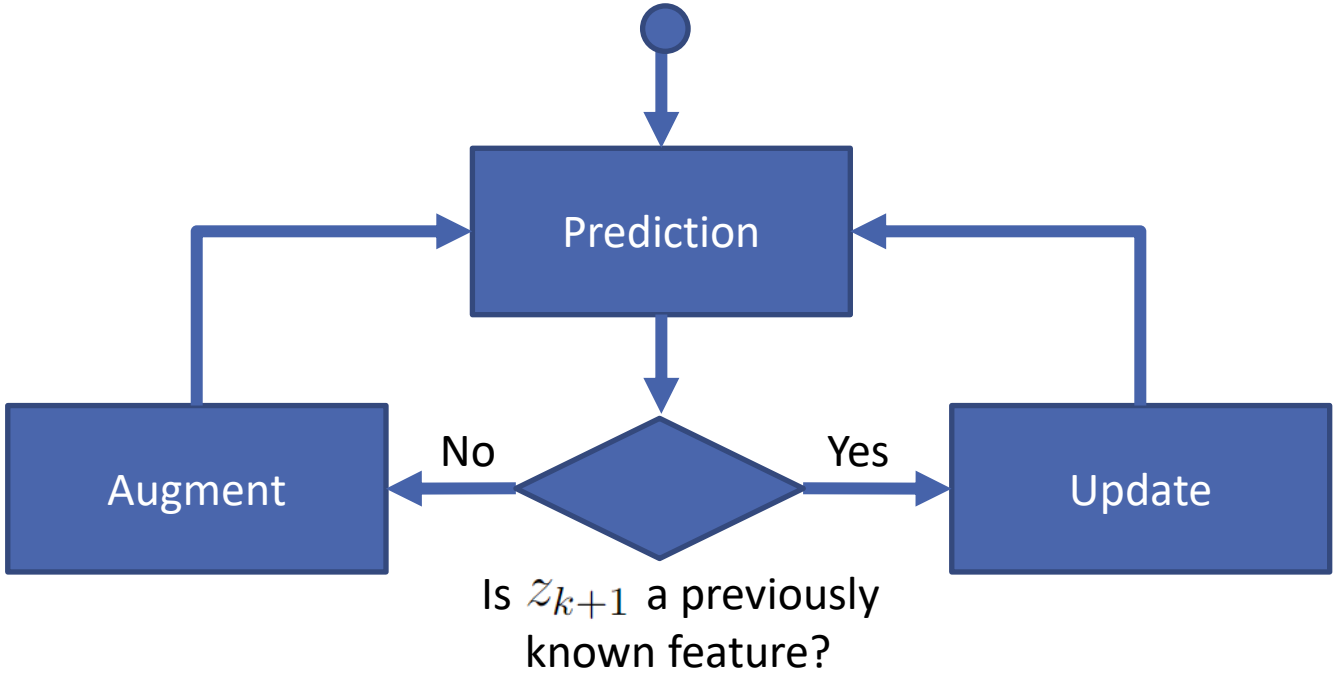
$$P = \begin{bmatrix} P_{vv} & P_{vm} \\ P_{mv} & P_{mm} \end{bmatrix}$$

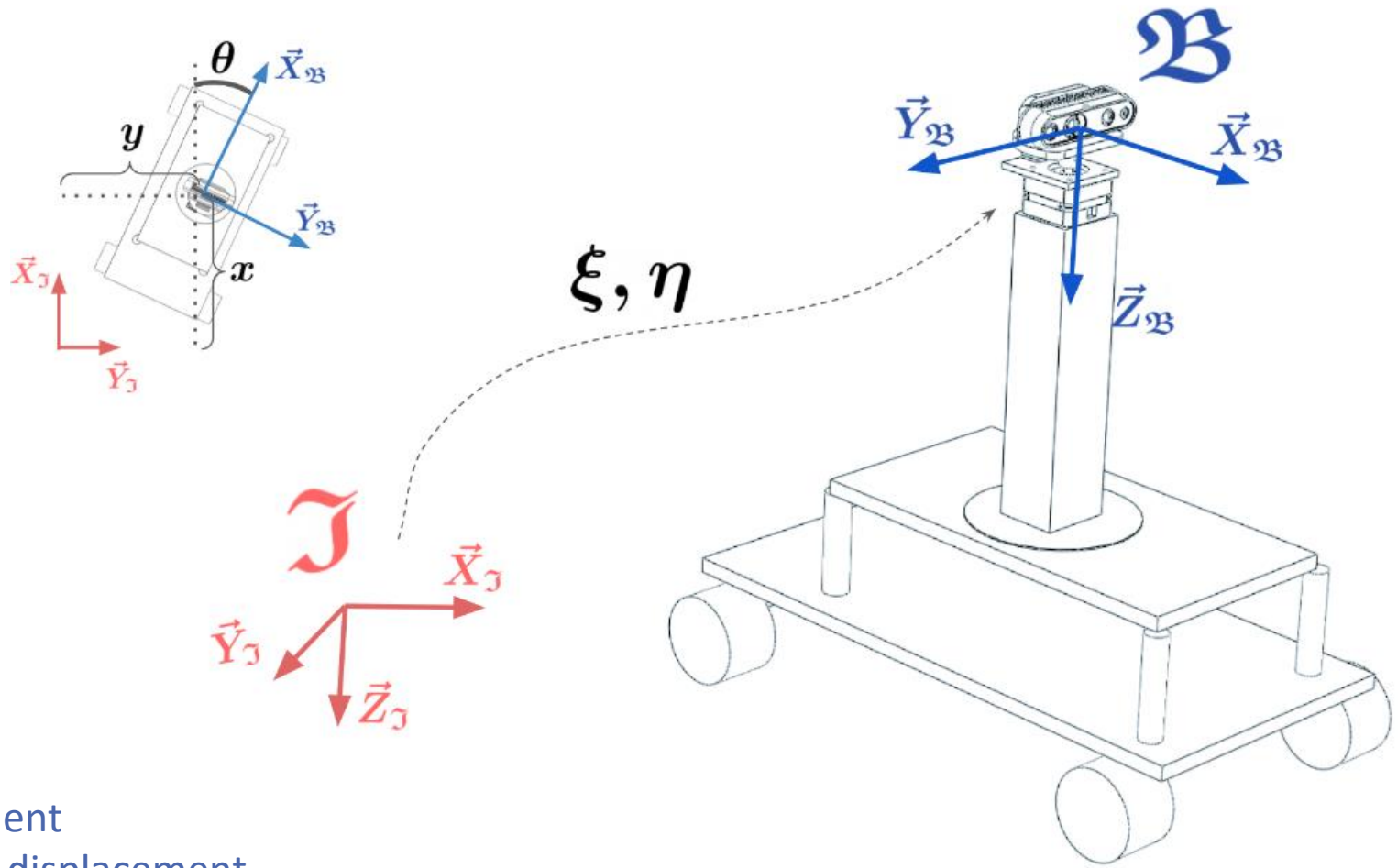
New feature detection

State vector augmentation:

$$\hat{x}^+_{k+1} = \begin{bmatrix} \hat{x}^+_{k+1} \\ g(\hat{x}^+_{k+1}, z_{k+1}) \end{bmatrix}$$

$$P^+_{k+1} = Y_z \begin{bmatrix} P^+_{k+1} & 0 \\ 0 & W_k \end{bmatrix} Y_z^T$$





Assumptions

- Planar movement
- Unidirectional displacement
- Z rotation

Process model

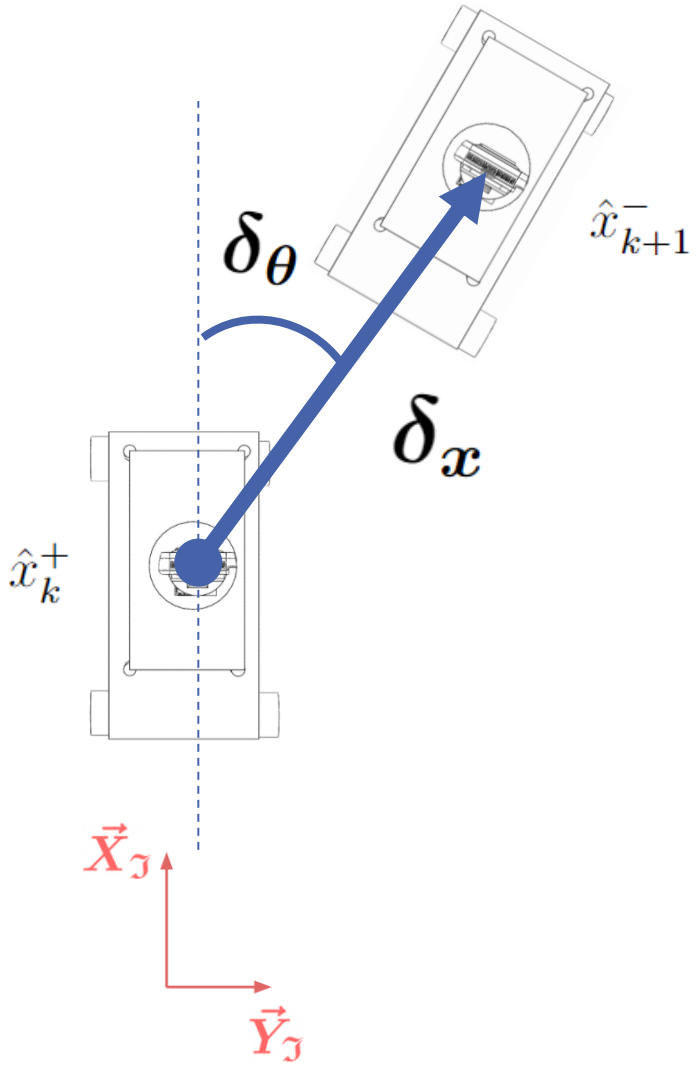
$$X_{k+1} = f(X_k, u_k, v_k) = \begin{bmatrix} x_k + (\cos\theta_k)(\delta_x + v_x) \\ y_k + (\sin\theta_k)(\delta_x + v_x) \\ \theta_k + (\delta_\theta + v_\theta) \end{bmatrix}$$

where:

$$X_k = [x_k \quad y_k \quad \theta_k]^T$$

$$u_k = [\delta_x \quad \delta_\theta]^T$$

$$v_k = [v_x \quad v_\theta]^T$$



Map representation

Plane segment → Plane
Cylinder → Point

(Parameterized map) (EKF)

Point feature

Cylinder centroid:

$$C = [c_x, c_y, c_z]^T$$

Measurement model:

$$z_{c\,i} = h_c(\hat{x}_k) = R^T_{zyx}\{\eta_k\}(C_{i,k} - \xi_k) + w_k$$

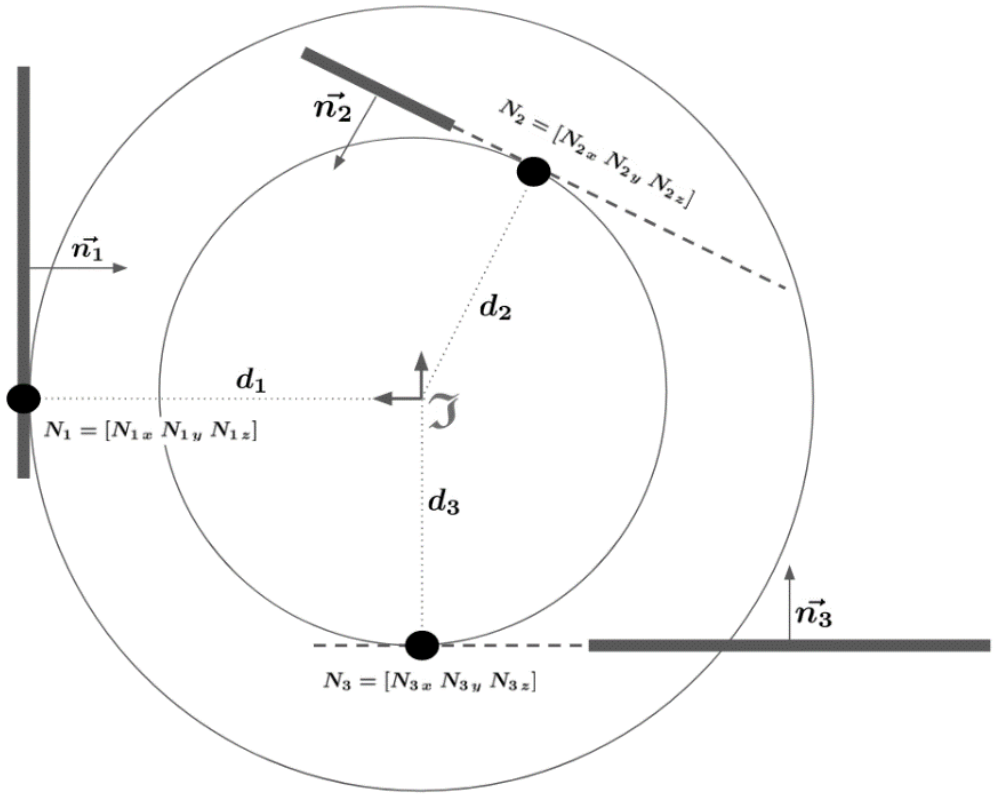
Plane feature

$$Ax + By + Cz + D = 0 \longrightarrow n_a x + n_b y + n_c z + d = 0$$

$$\vec{n} = [n_a, n_b, n_c]$$

Compact plane representation:

$$N = d.\vec{n} = [N_x \quad N_y \quad N_z]^T$$

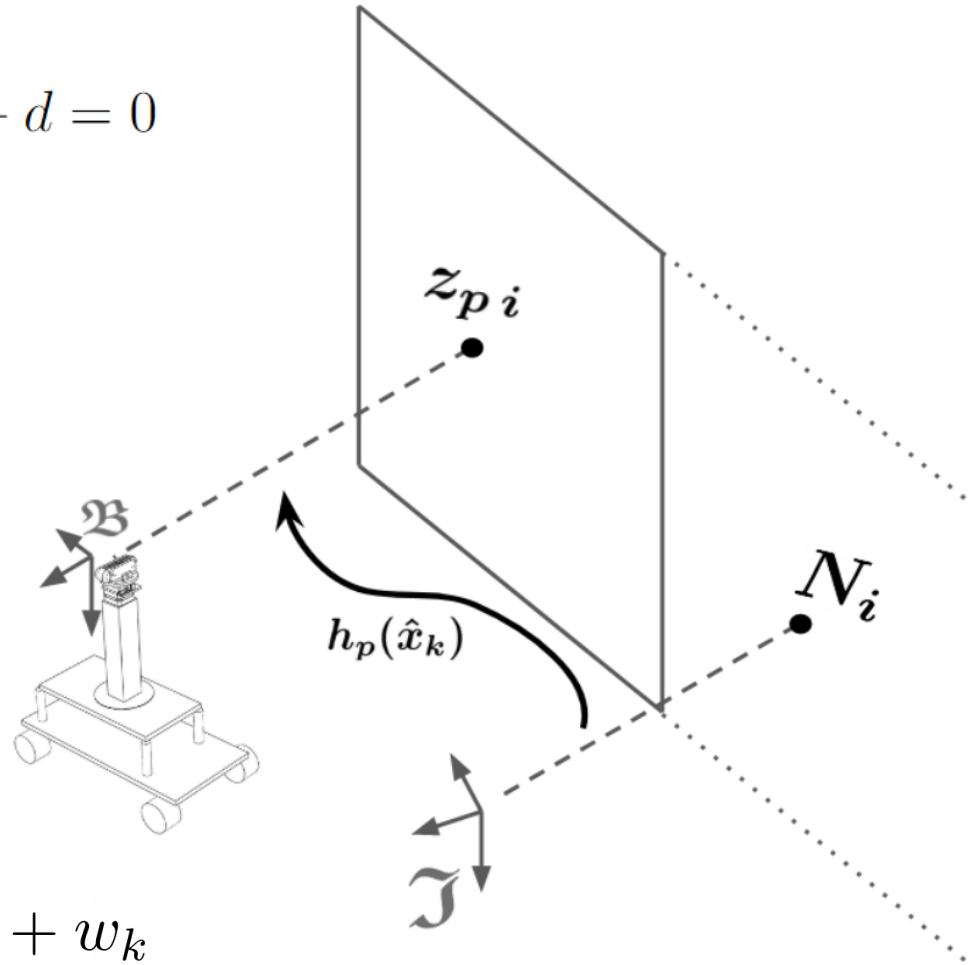


Plane feature

$$Ax + By + Cz + D = 0 \longrightarrow n_a x + n_b y + n_c z + d = 0$$

Compact plane representation:

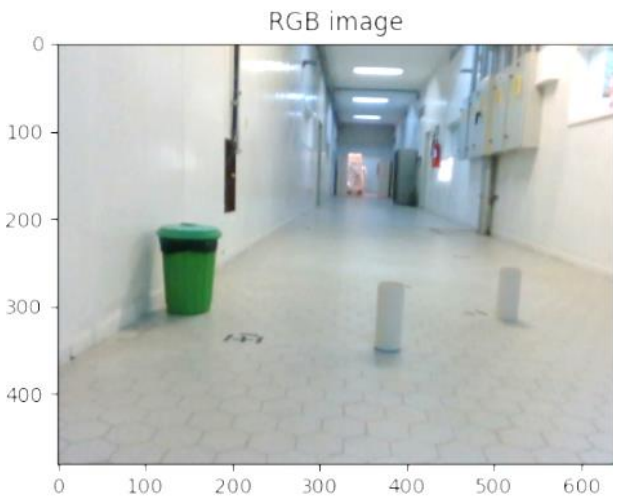
$$N = d.\vec{n} = [N_x \quad N_y \quad N_z]^T$$

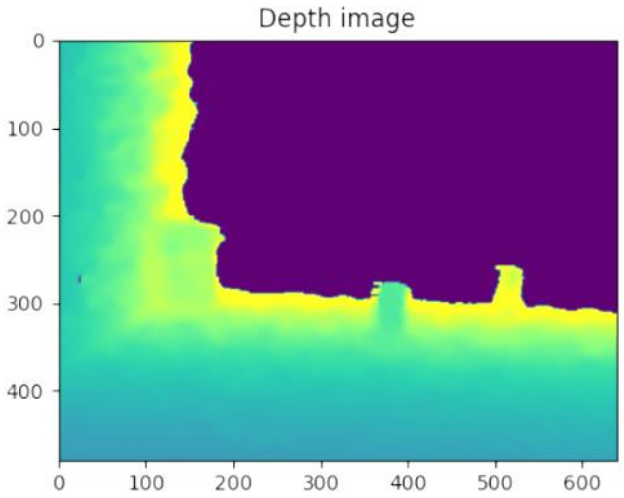


Measurement model:

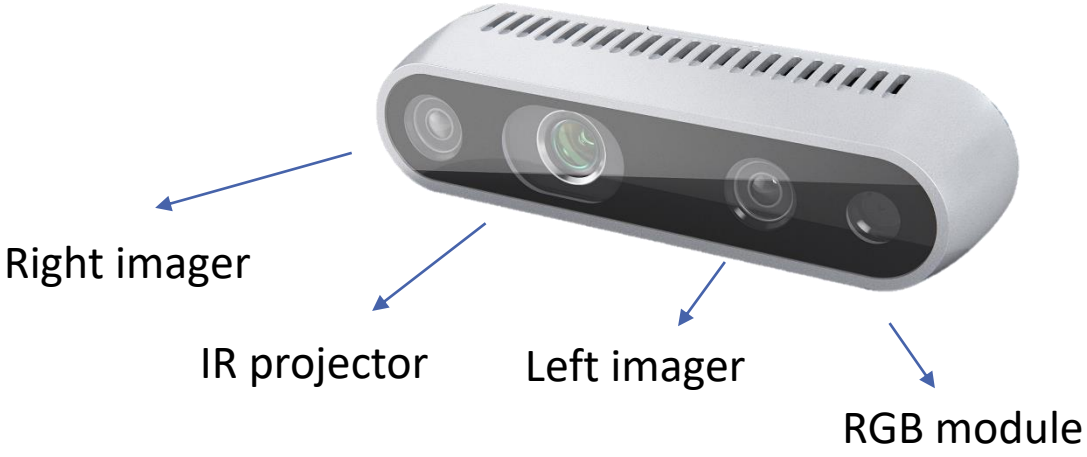
$$z_{p\,i} = h_p(\hat{x}_k) = R^T_{zyx}\{\eta_k\}(N_{i,k} + \frac{\xi_k \cdot N_{i,k}}{||N_{i,k}||^2} N_{i,k}) + w_k$$

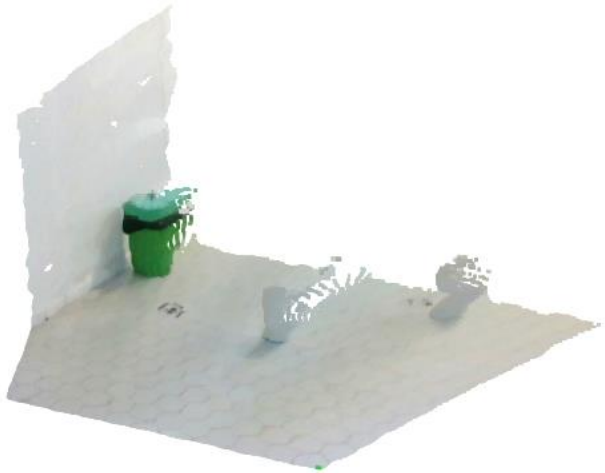
Feature extraction





Intel RealSense Depth Camera D435i





Input

Plane segment extraction

Plane parameters

Point clouds

Cylinder extraction

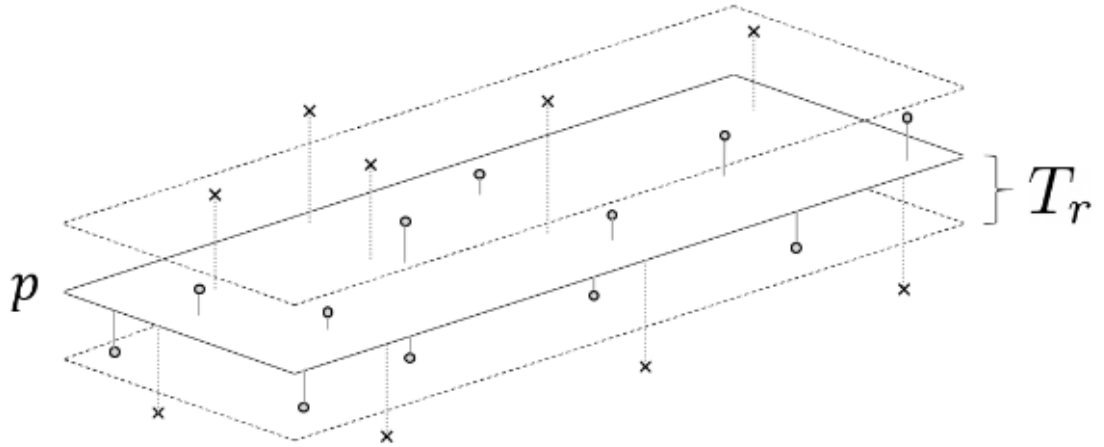
Cylinder parameters

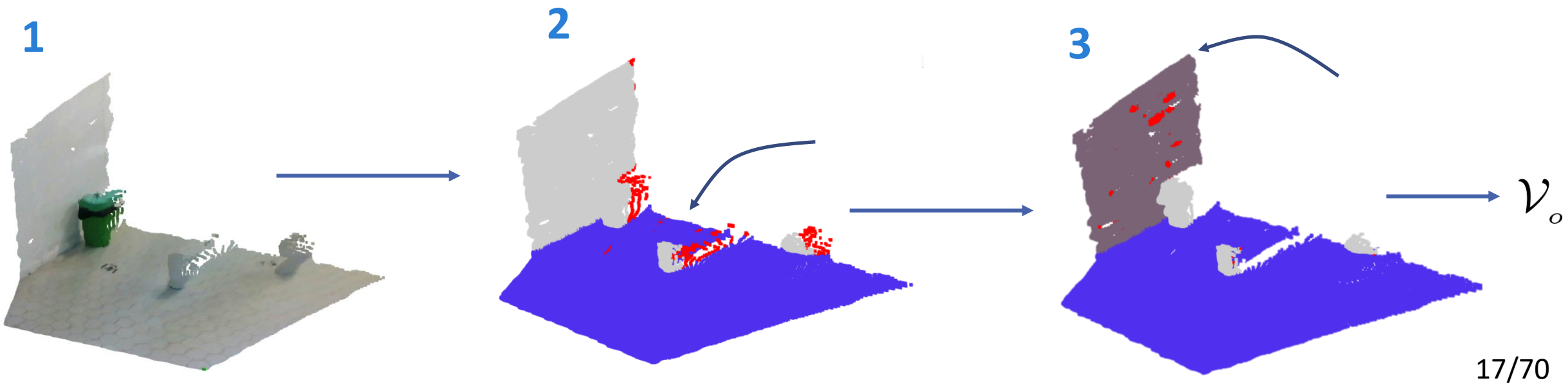
Point clouds

Plane extraction

Planar RANSAC

- Robust against noise





Plane extraction

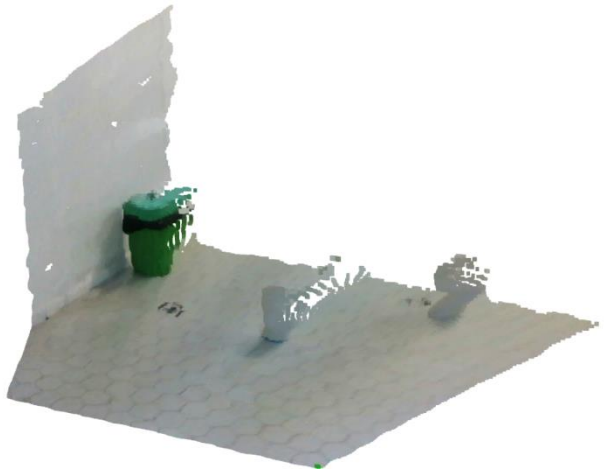
Plane segment parameterization

Point cloud + plane eq.

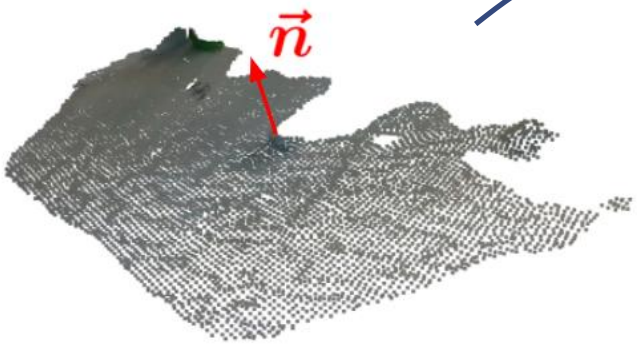
2D projection

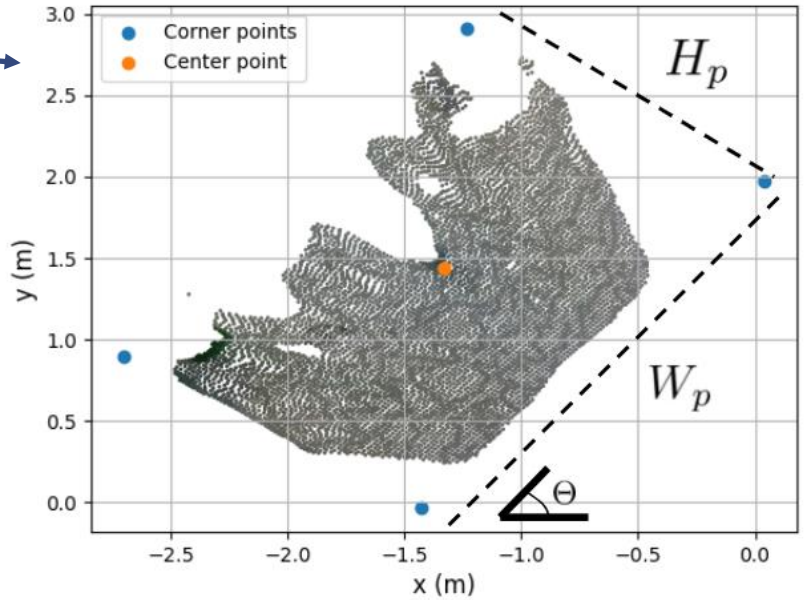
Minimum area rectangle (MAR)

H_p, W_p, c_{2d}, Θ

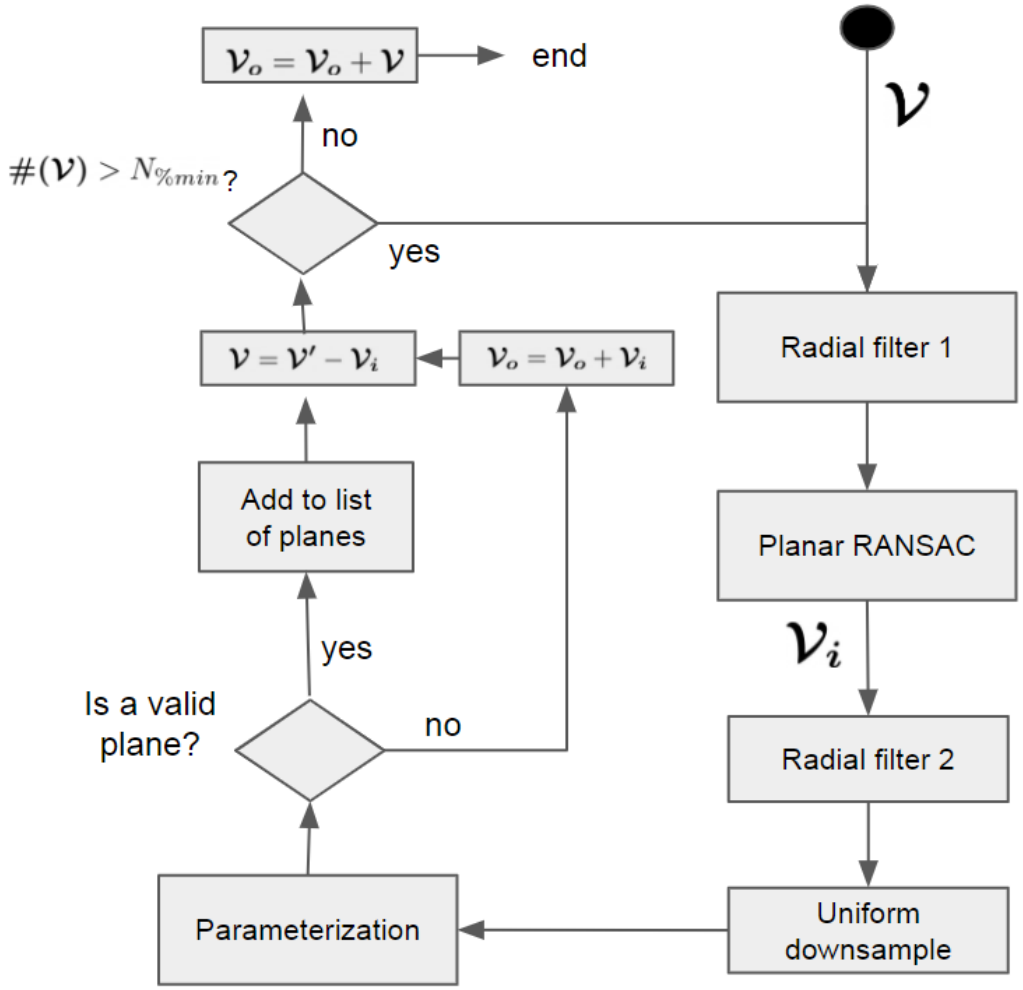


RANSAC

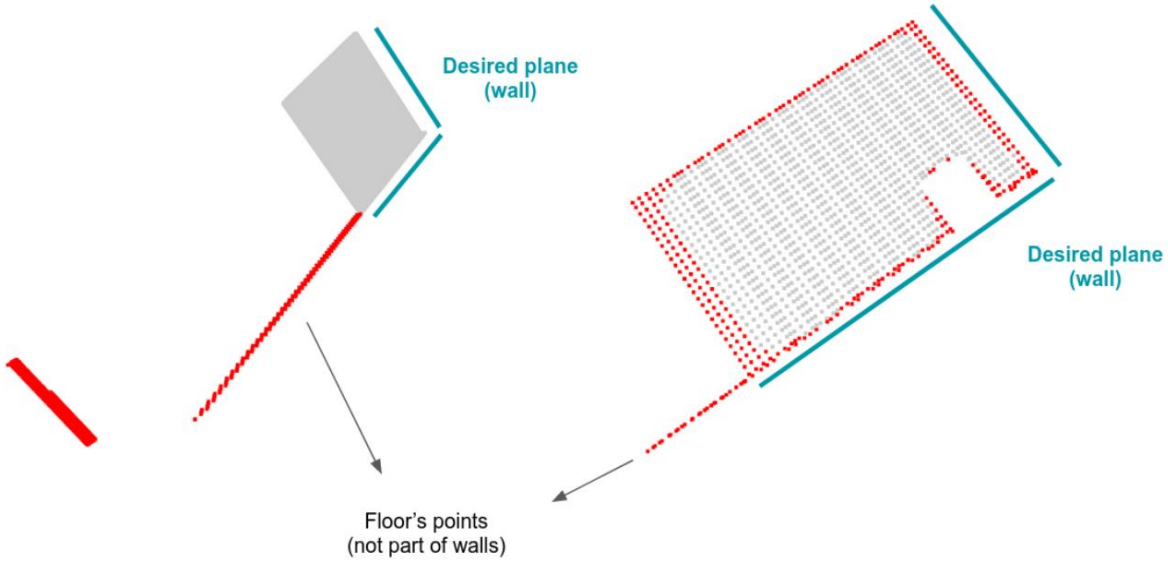




Plane extraction



Radial filter 2:



Validation gates:

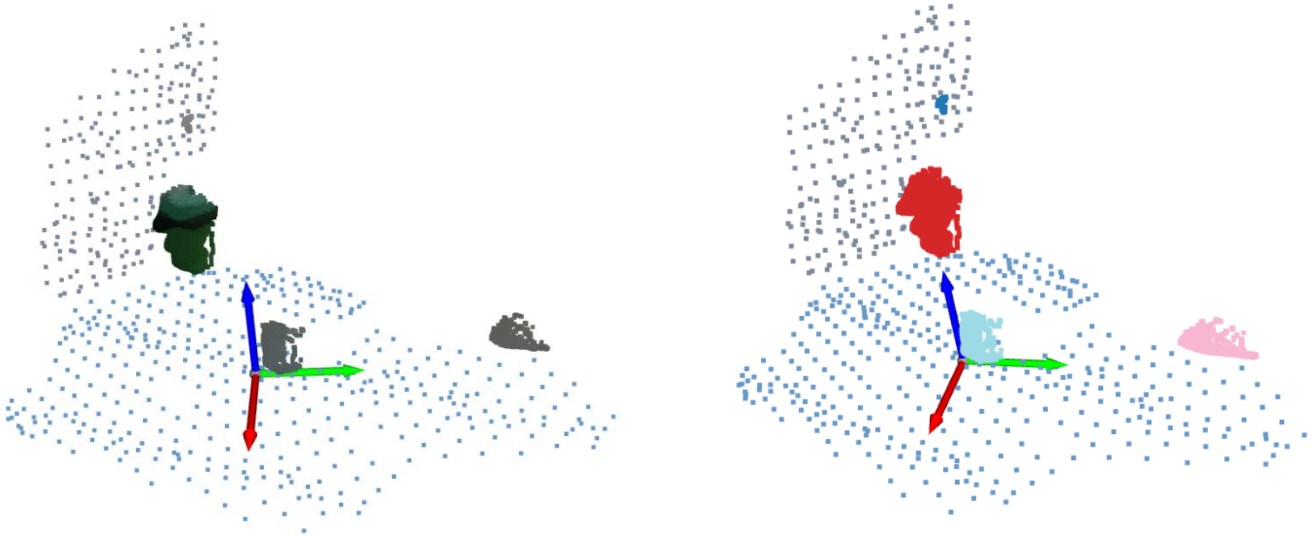
- Centroid
- Density
- Area

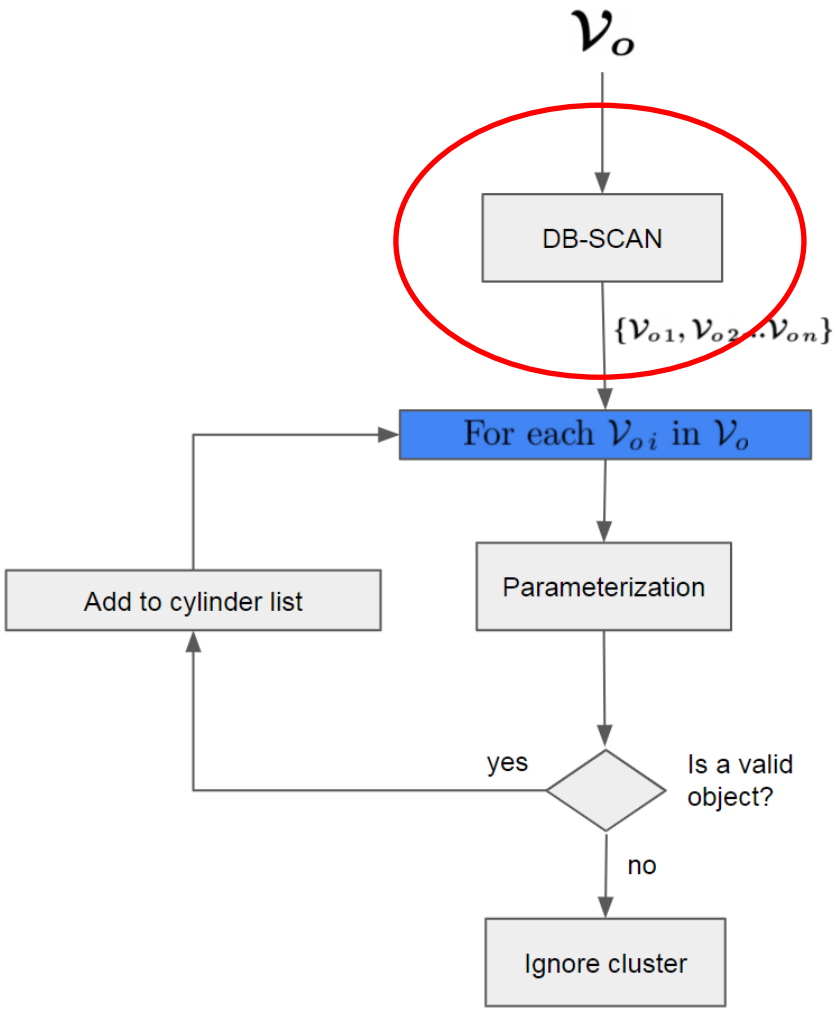
Cylinder extraction

Assumptions:

- Axis aligned to the floor's normal, no oblique or tilted cylinder
- Object inside the bounds of the image

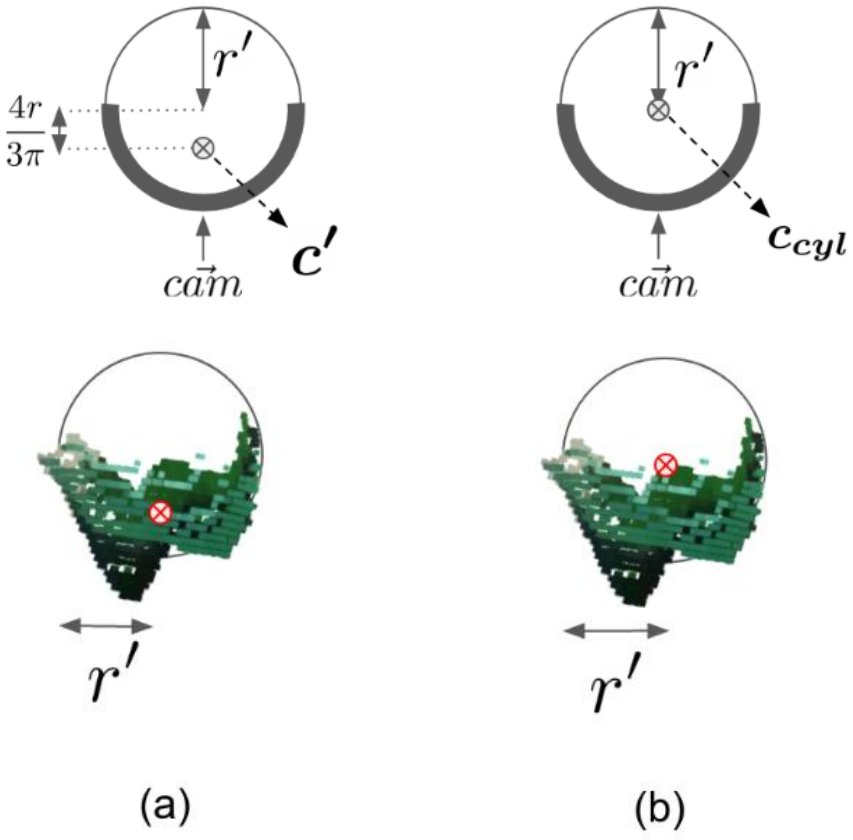
Clusterization:

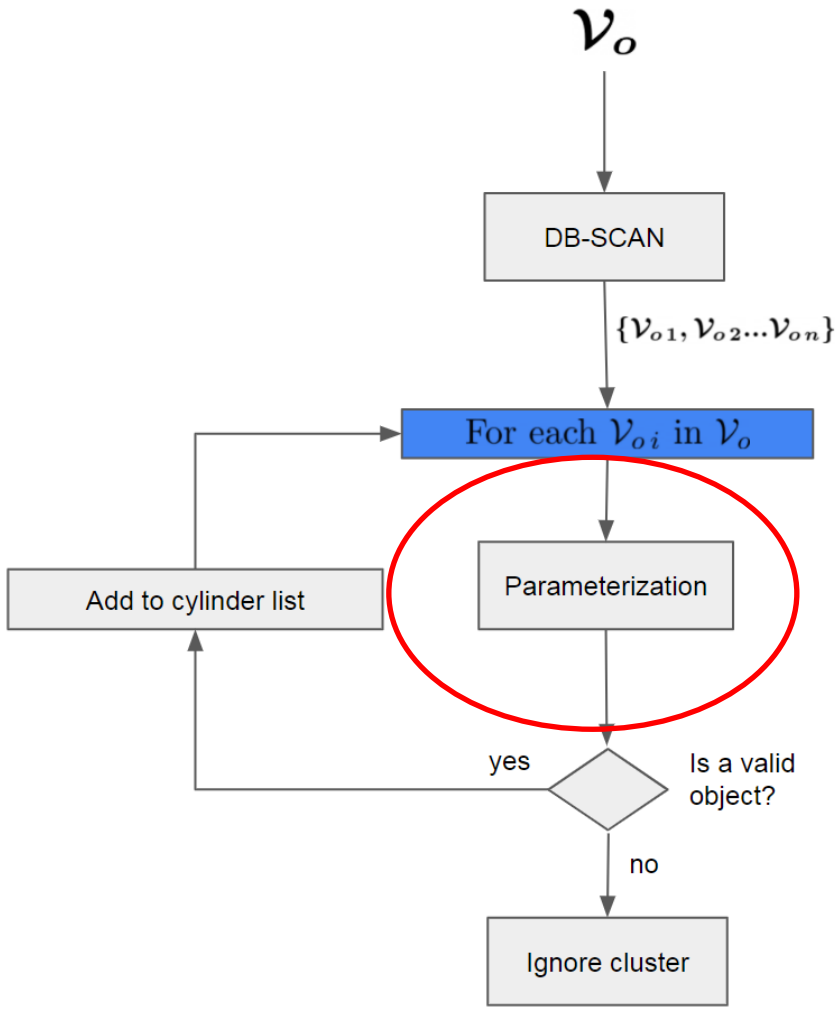




Cylinder extraction

Cylinder parameterization: $H_{cyl}, r_{cyl}, c_{cyl}$



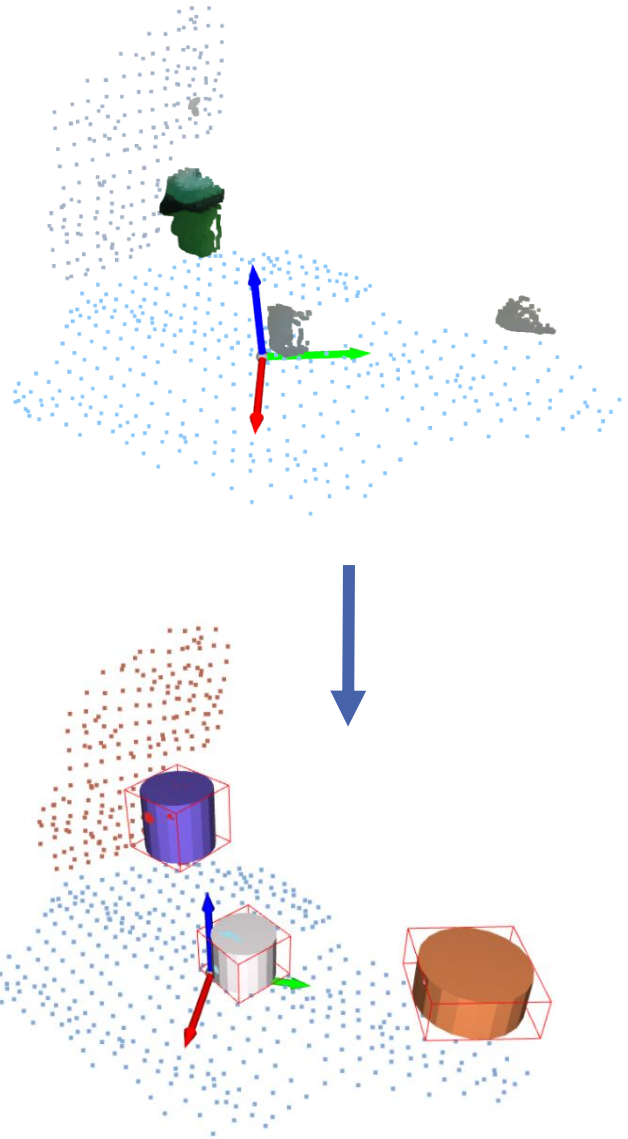


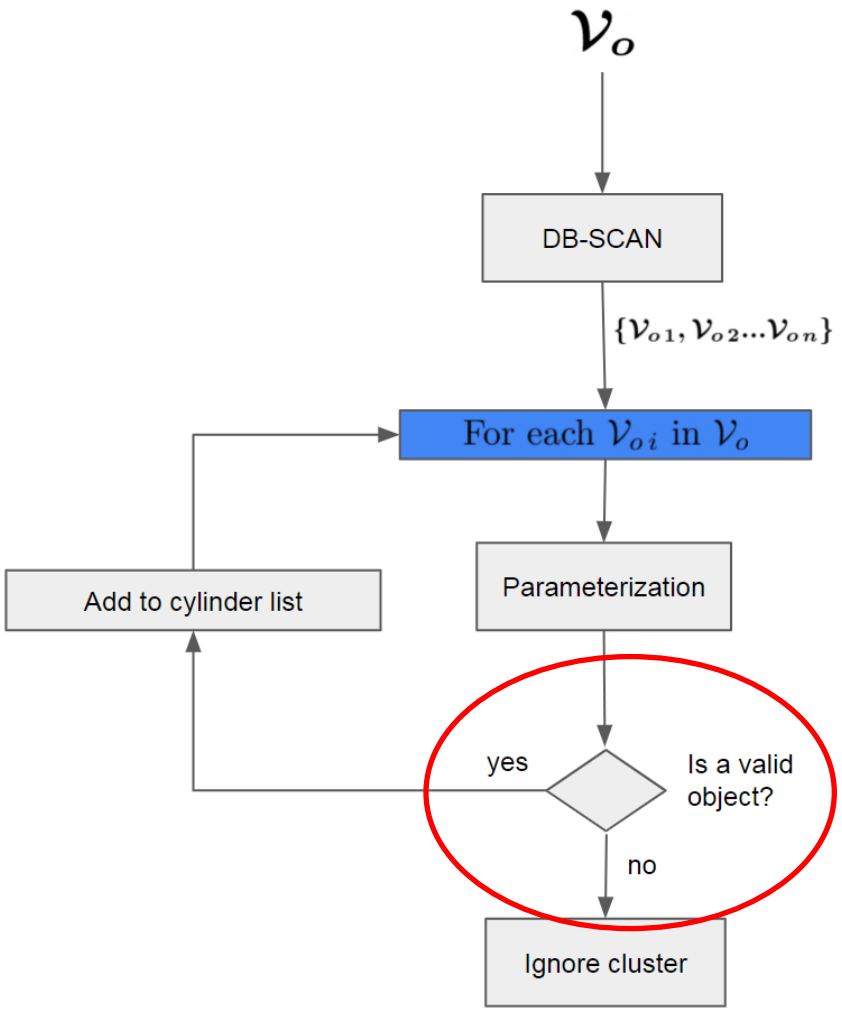
Cylinder extraction

Validation gates:

- Radius
- Radius noise

$$\frac{\sigma(\ \mathcal{R}_{cyl}\ )}{\mu(\ \mathcal{R}_{cyl}\ )} < \tau_{max}$$

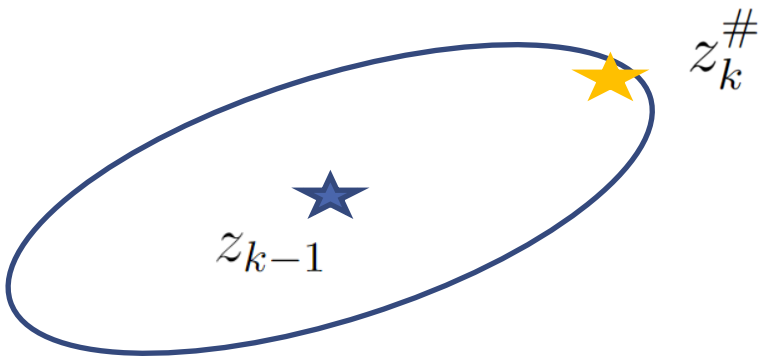


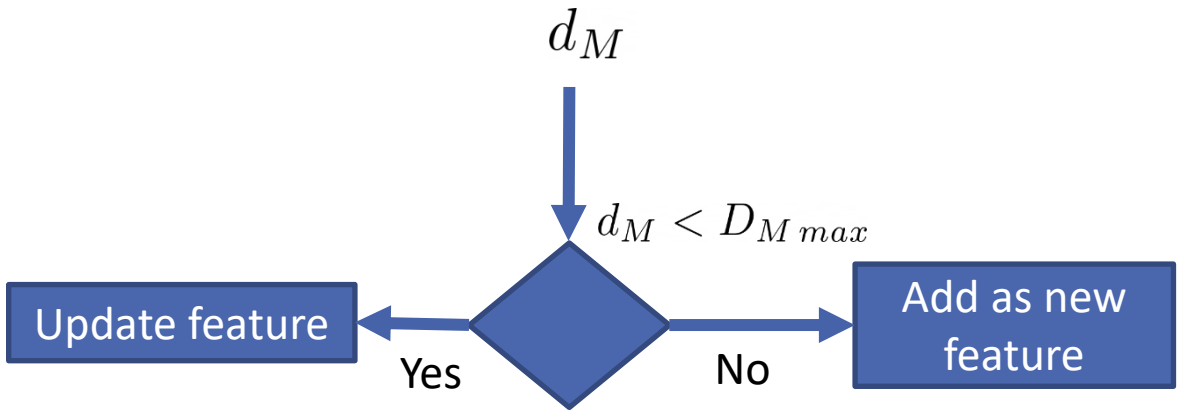


Data association

Mahalanobis distance:

- Accounts for covariance.
- Scale invariant.



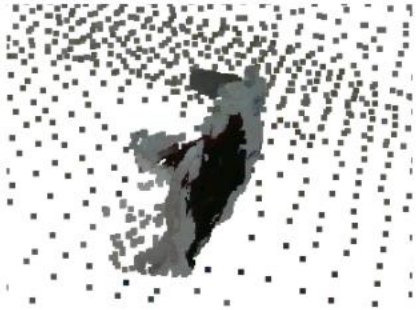


$$d_M = \sqrt{(z_k^{\#} - z_{k-1})^T S^{-1} (z_k^{\#} - z_{k-1})}$$

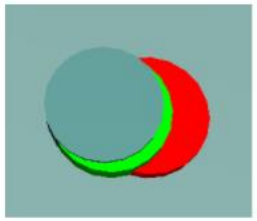
$$S = H_x P H_x^T + H_w W H_w^T$$

Feature growth

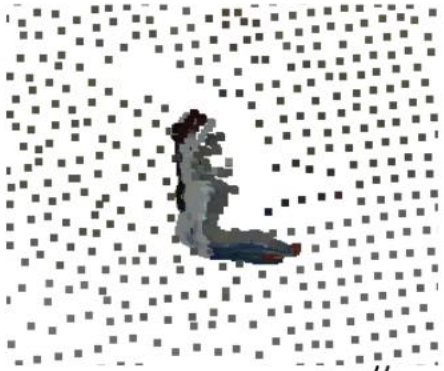
Cylinder growth



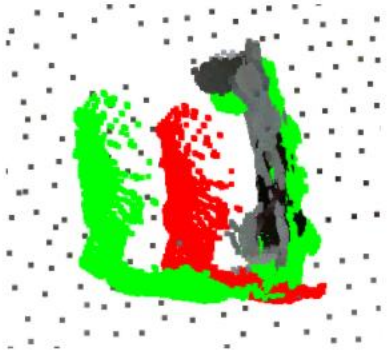
Previous point cloud $\mathcal{V}_{o\,k-1}$



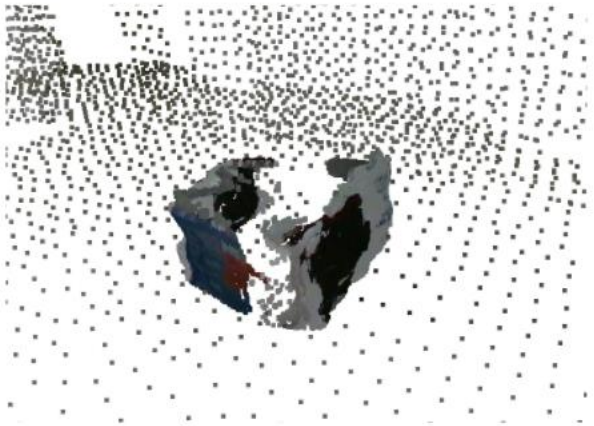
EKF Update



Measured point cloud $\mathcal{V}_o^{\#}$



Updated feature

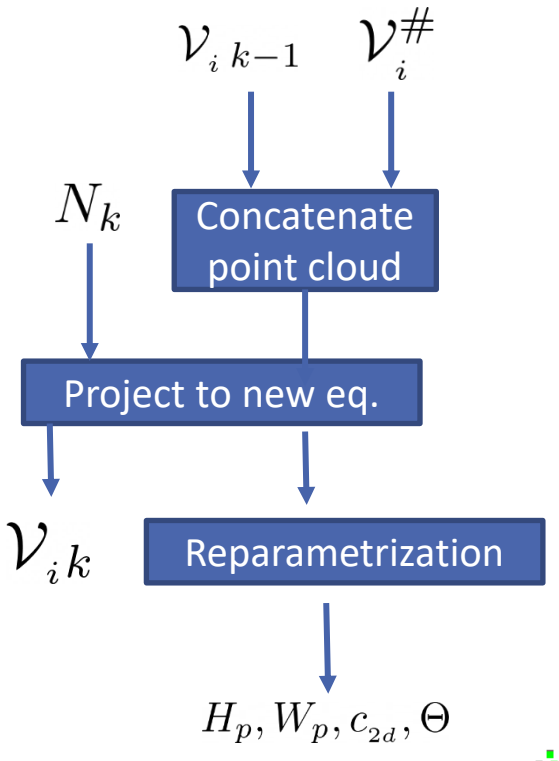


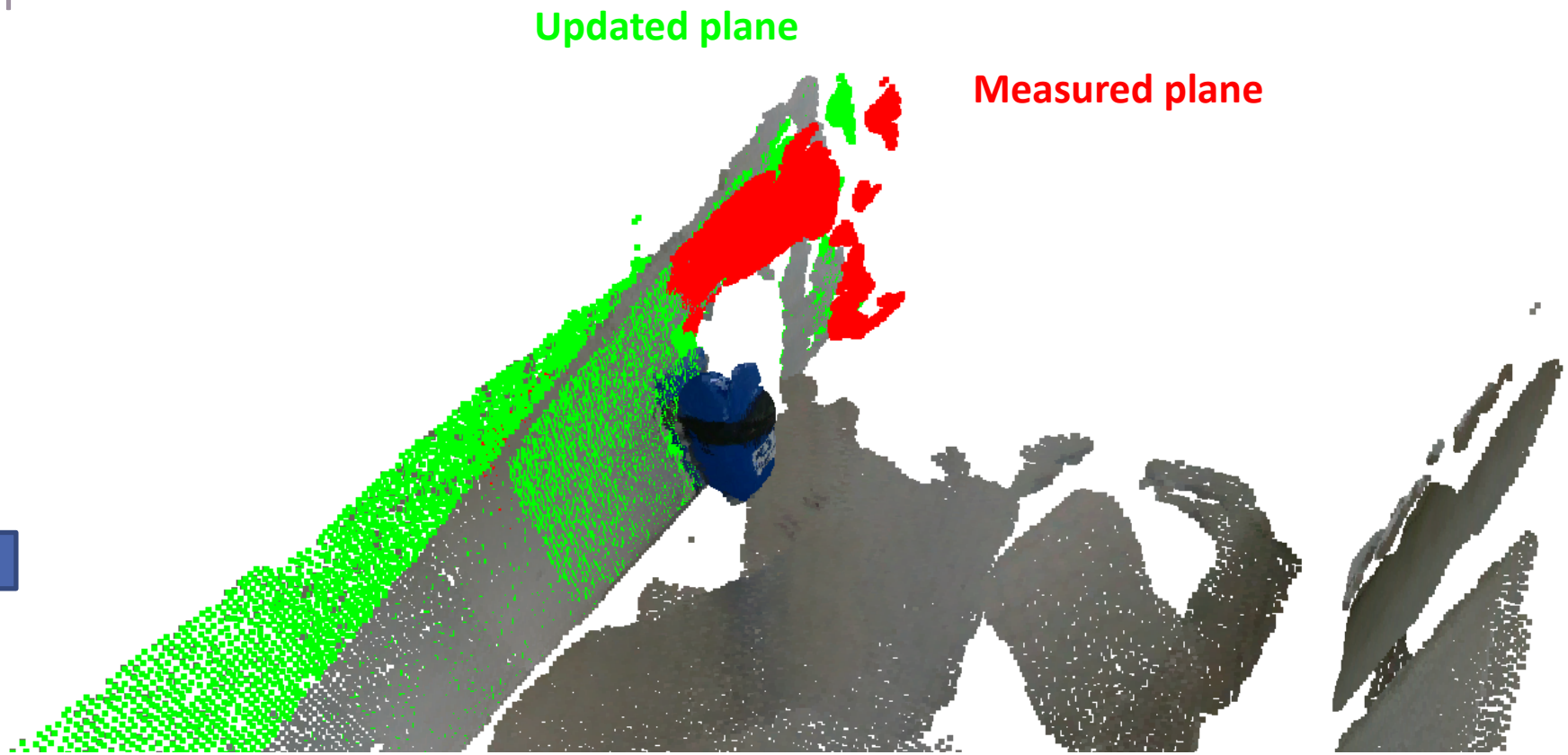
New point cloud $\mathcal{V}_{o\,k}$

Measured feature

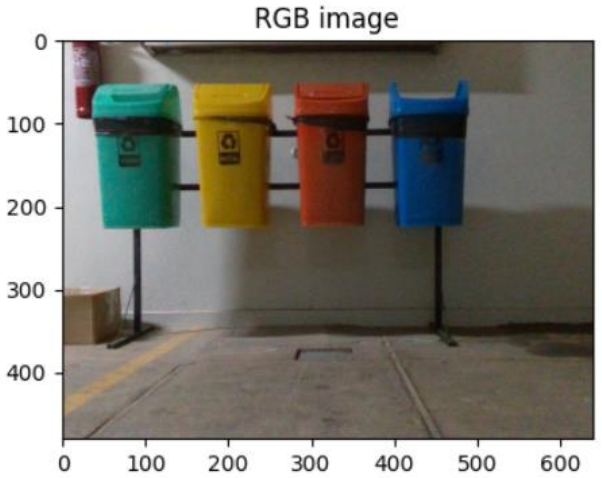
Feature growth

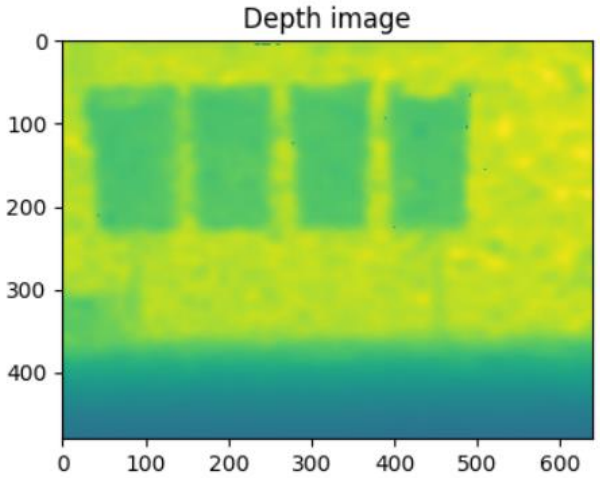
Plane segment growth



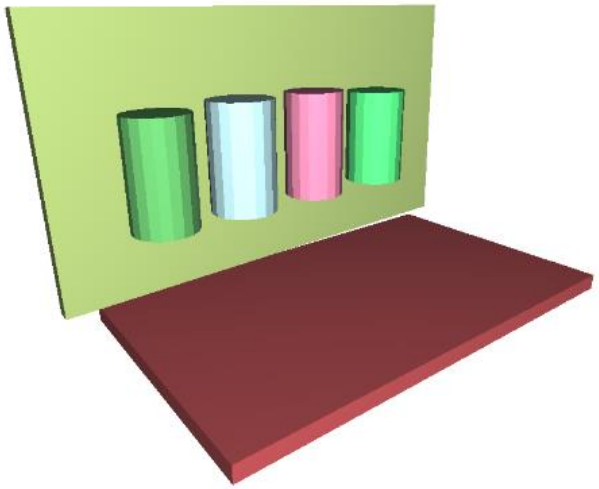


Map representations





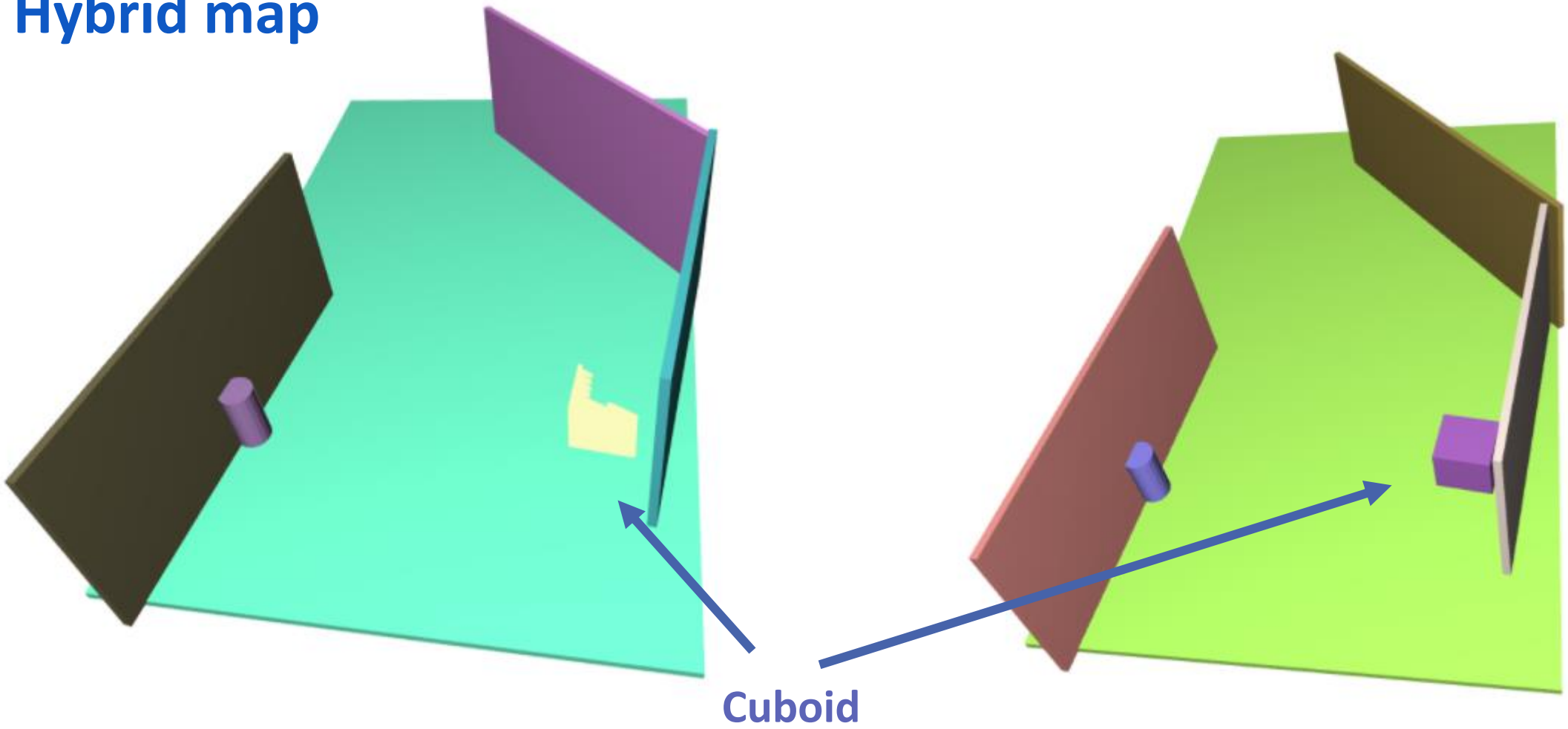
High level map



Point cloud map

Map representations

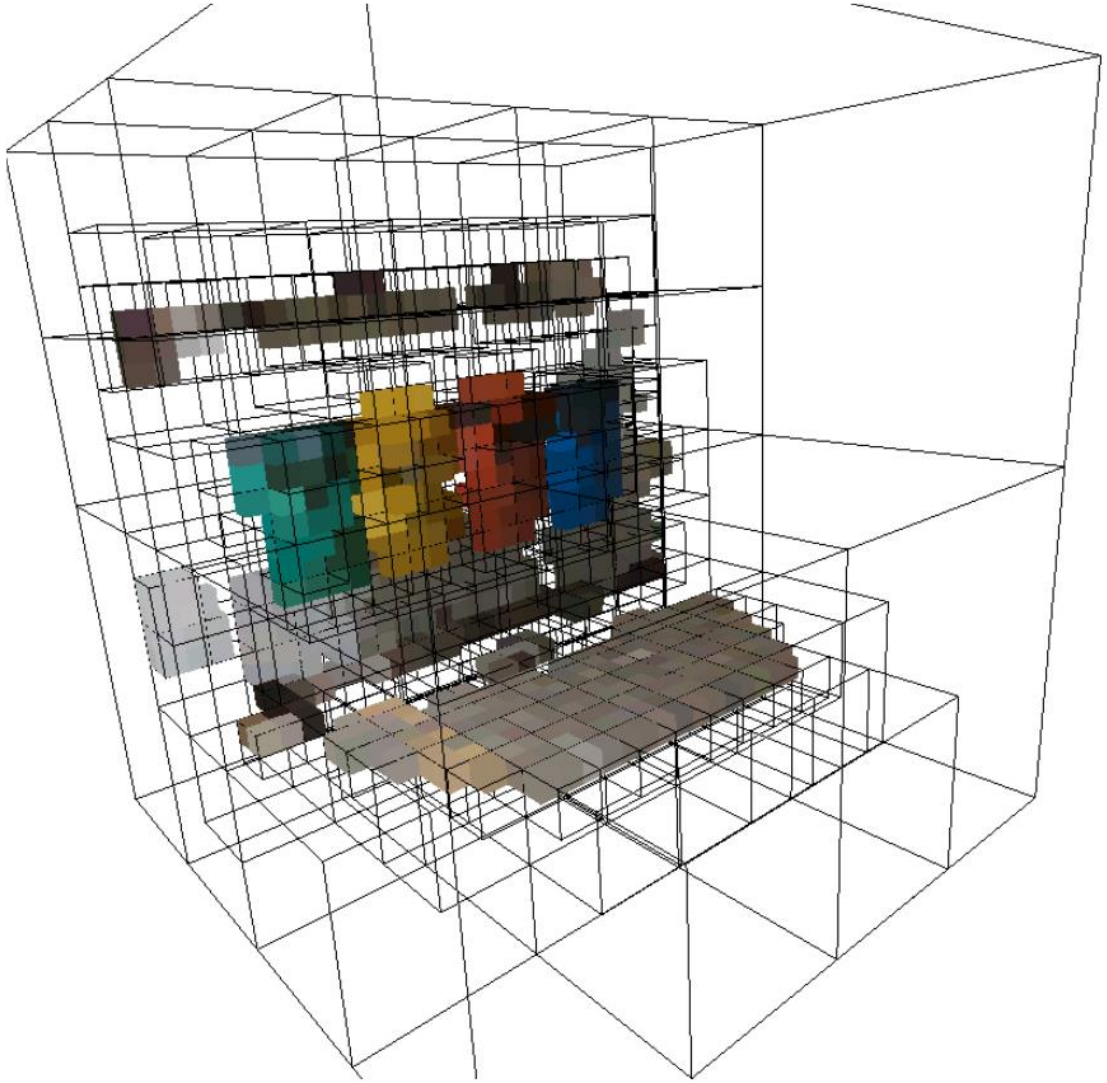
Hybrid map



Map representations

Voxel grid

Octree



Integration

Code implementation:

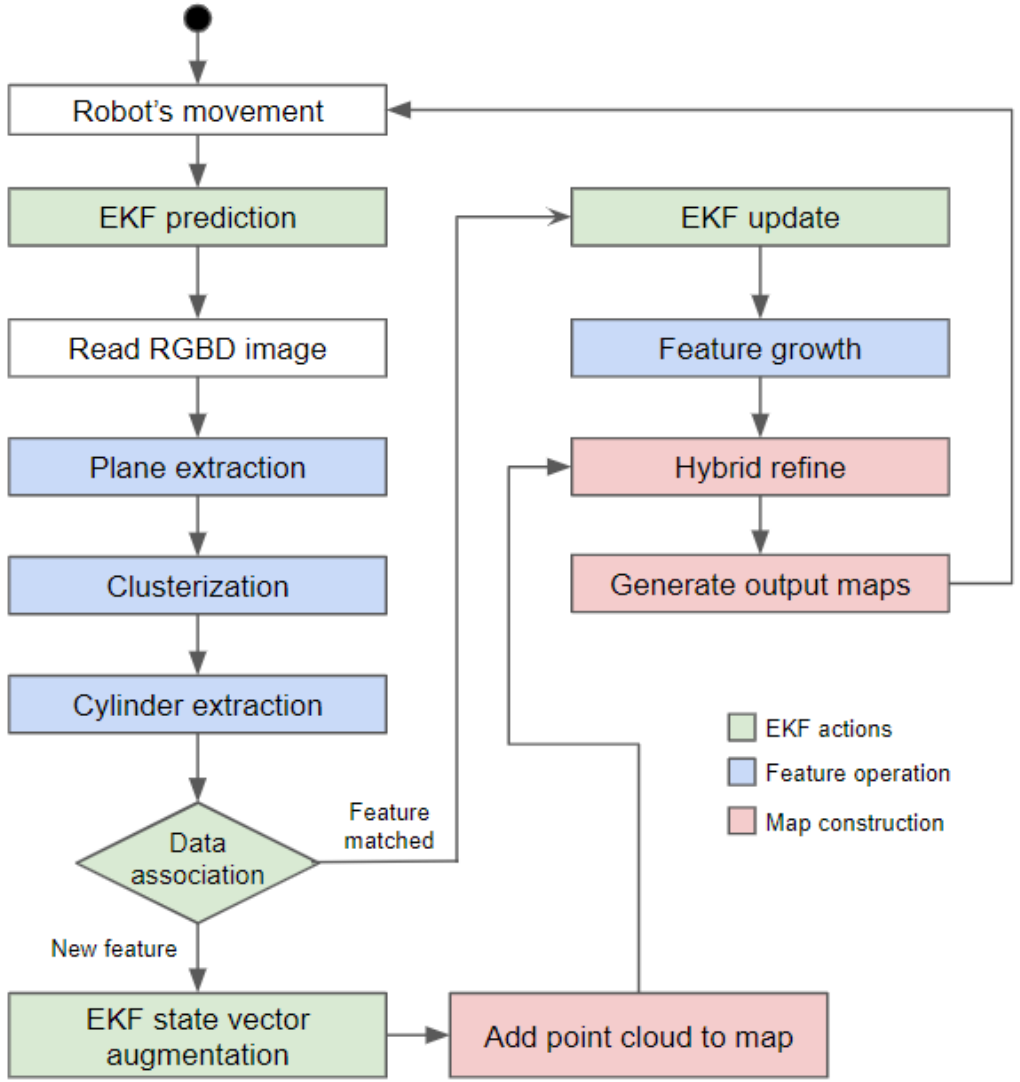
python 3.6

Main program

- EKF
- Feature extraction logic
- Data association
- Feature growth logic

Open3D

- Point cloud operations
- Map data structures
- 3D visualization

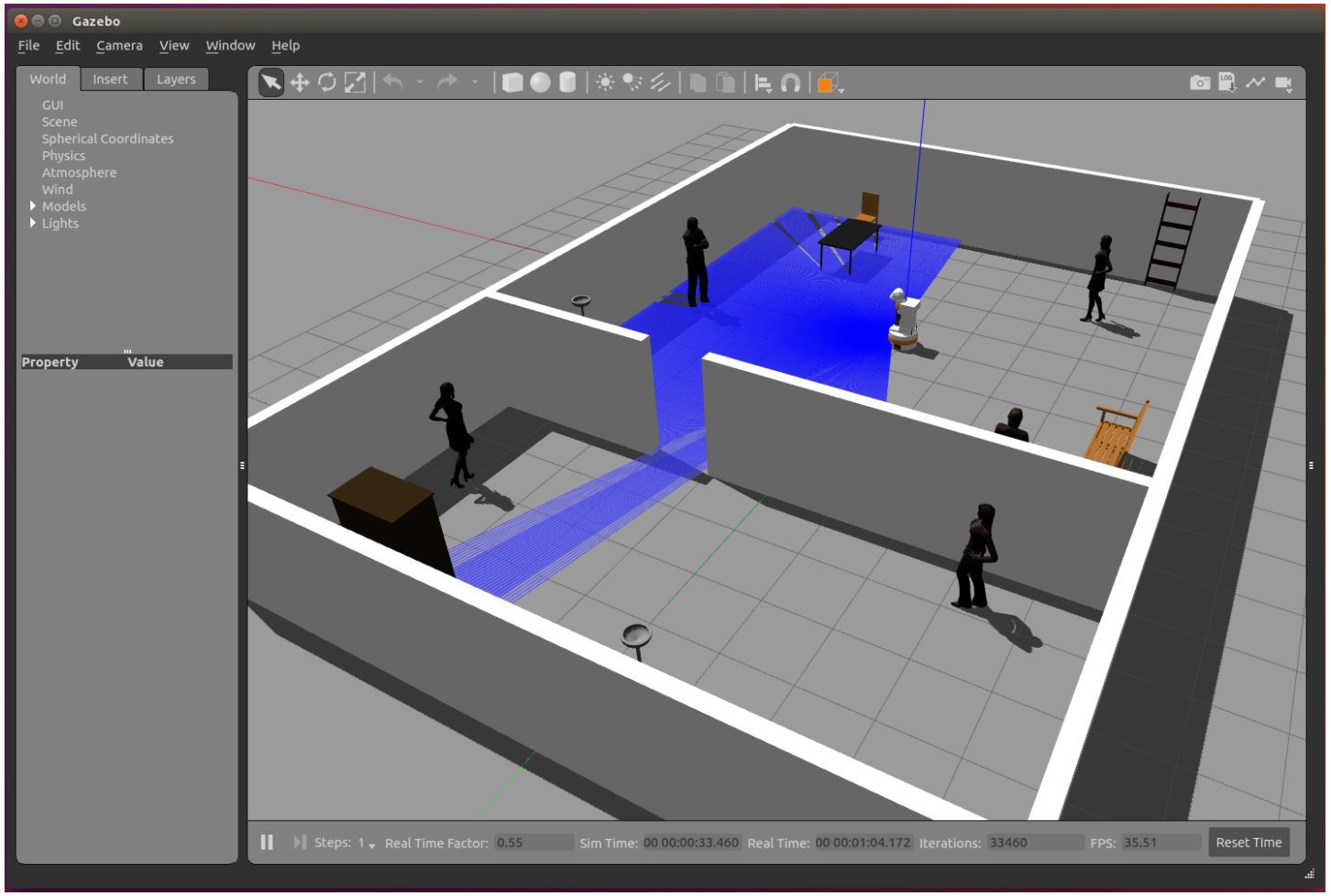


Other libraries: Matplotlib, Numpy, OpenCV

Simulation

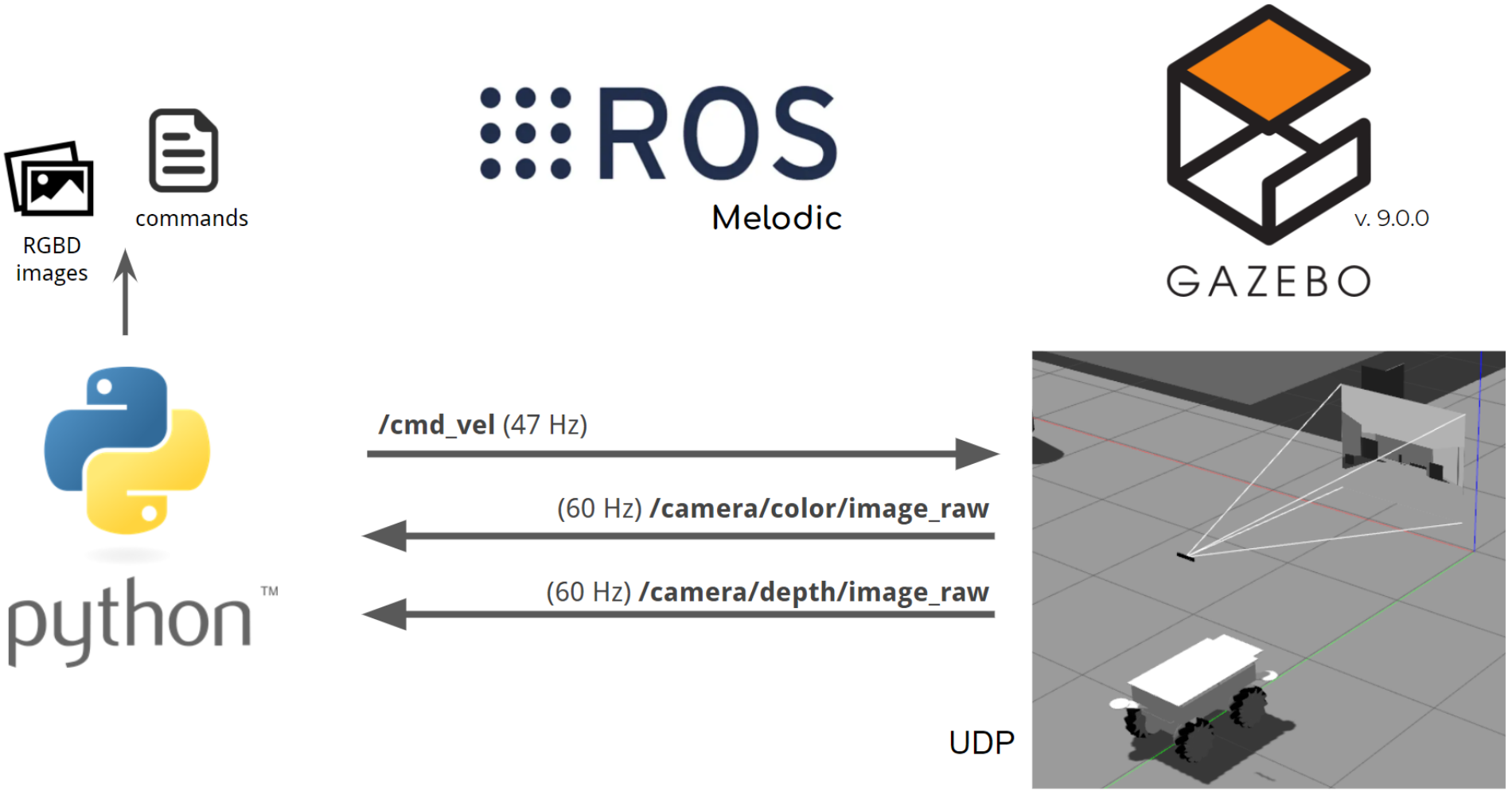
v. 9.0.0

GAZEBO



RobMoSys (2019)

Simulation


RGBD images
commands
ROS
Melodic
v. 9.0.0
GAZEBO
python
/cmd_vel (47 Hz)
(60 Hz) /camera/color/image_raw
(60 Hz) /camera/depth/image_raw
UDP

Simulation

$$u_k = [\delta_x \quad \delta_\theta]^T$$

Odometry → Command

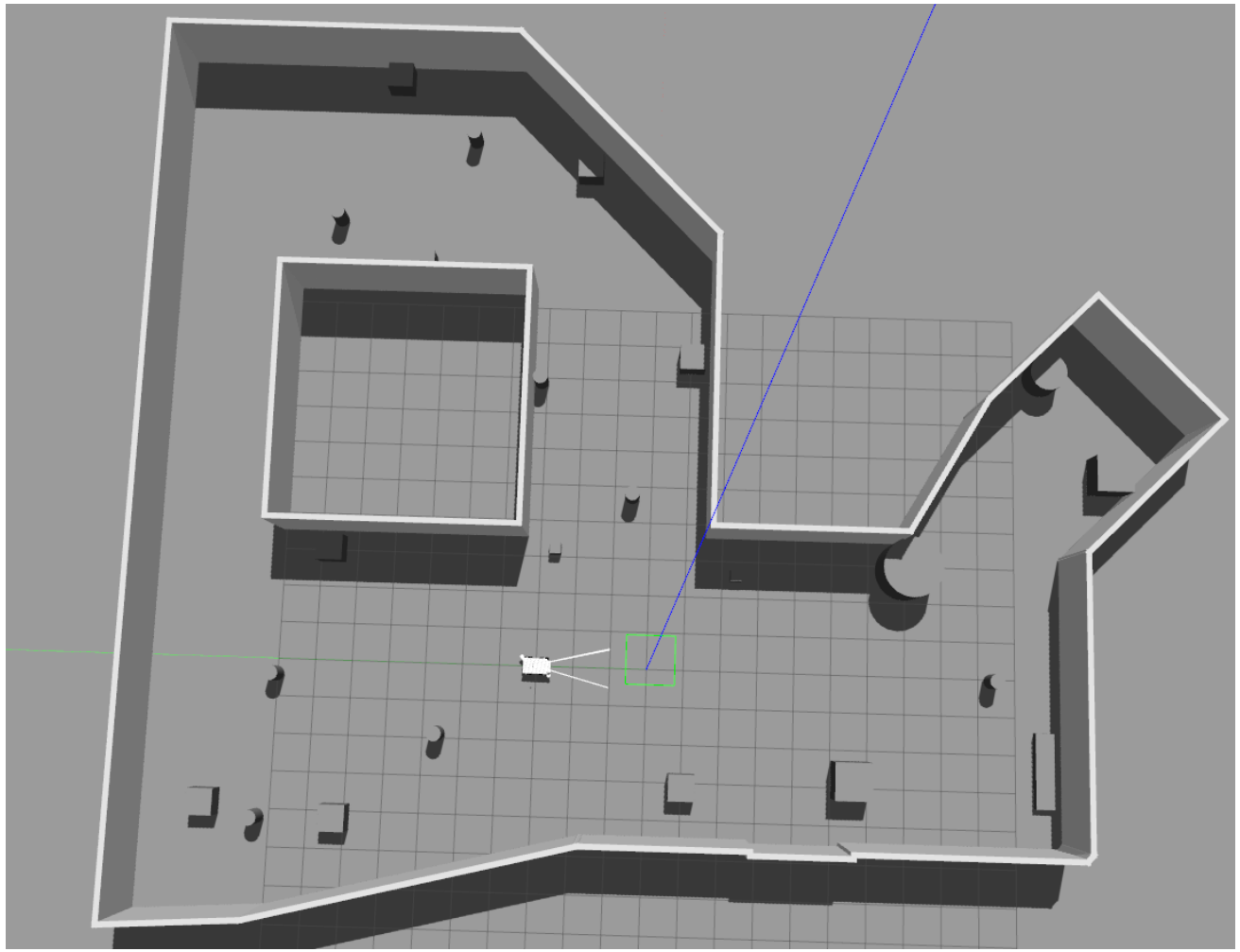
Step: 0.8 m, 45°

+

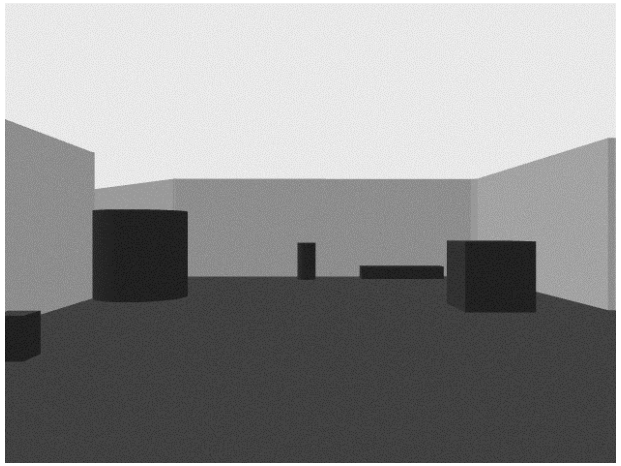
$$v_k \approx \mathcal{N}(0, V)$$

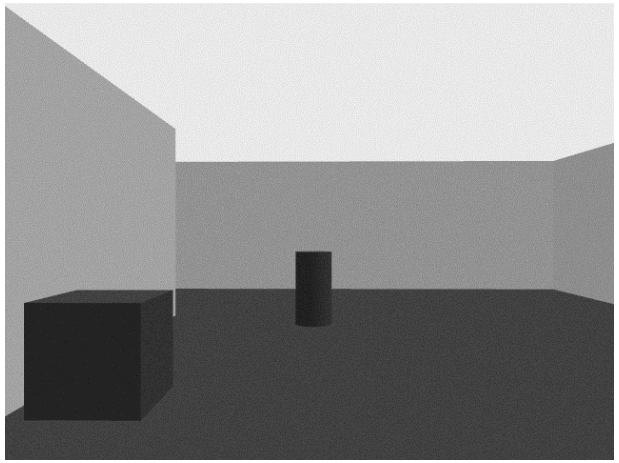
$$V = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \qquad \hat{V} = V$$

$$\hat{W} = \begin{bmatrix} \sigma_X^2 & 0 & 0 \\ 0 & \sigma_Y^2 & 0 \\ 0 & 0 & \sigma_Z^2 \end{bmatrix} \qquad \sigma_X = \sigma_Y = \sigma_Z = \frac{0.1}{3}$$

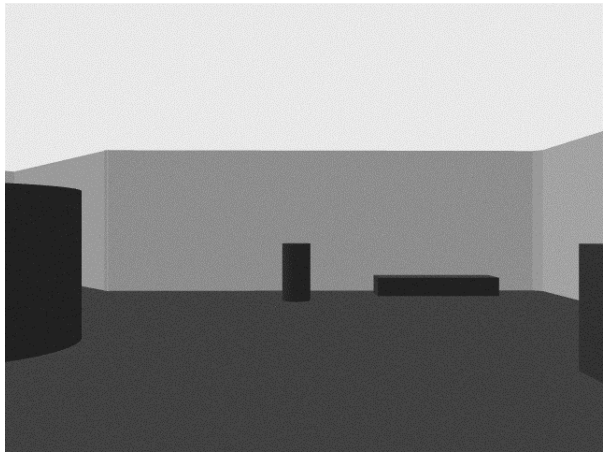


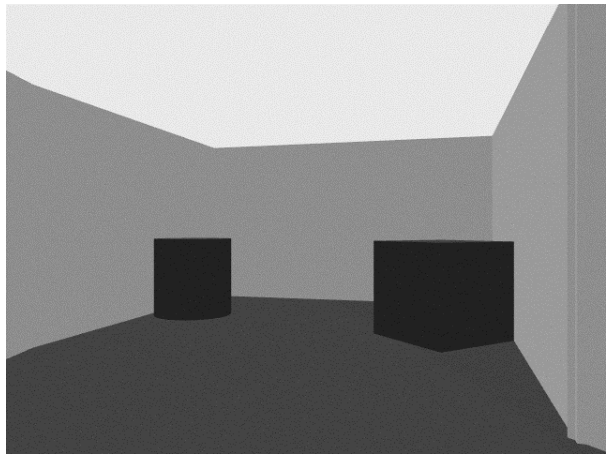
Simulation

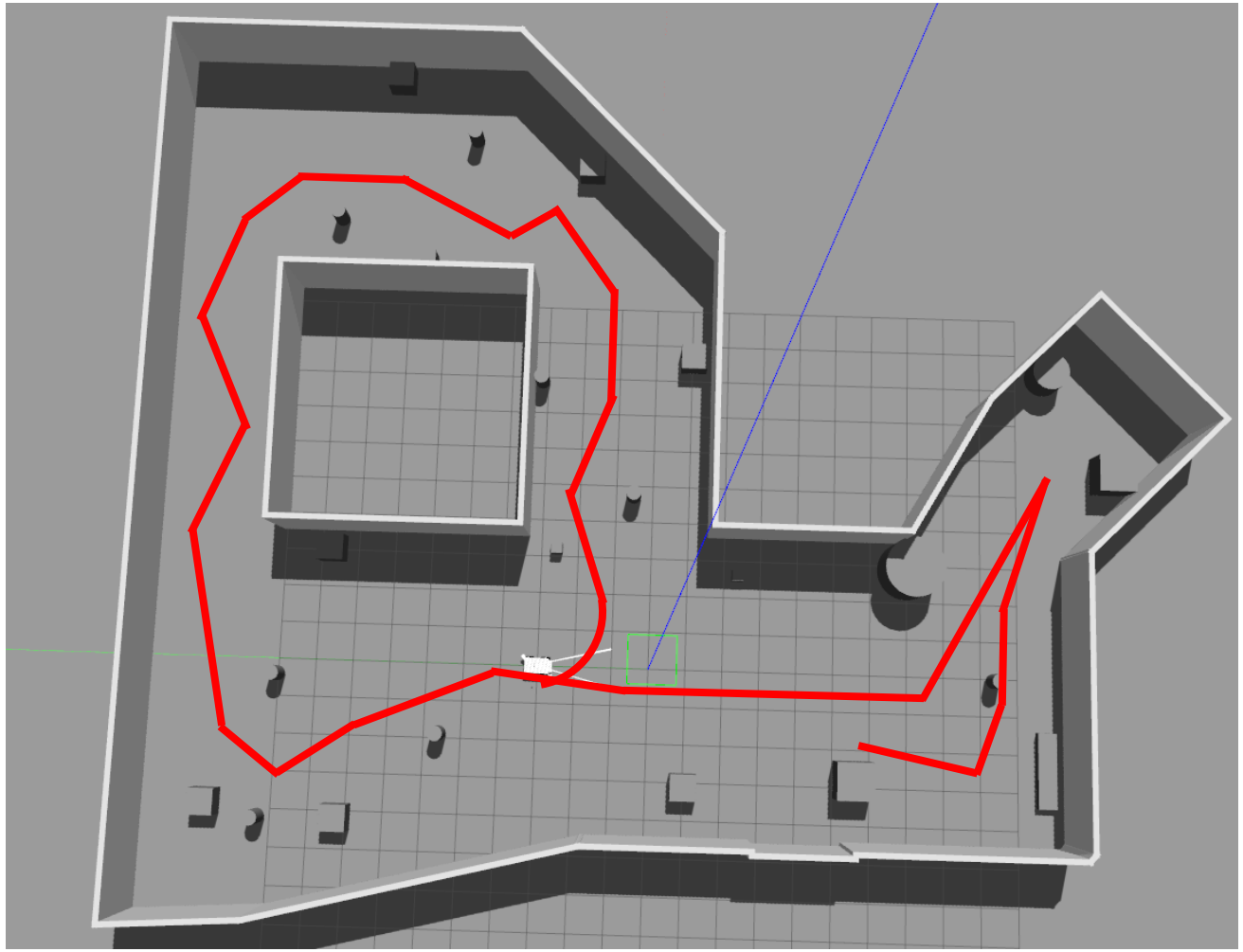




Simulation

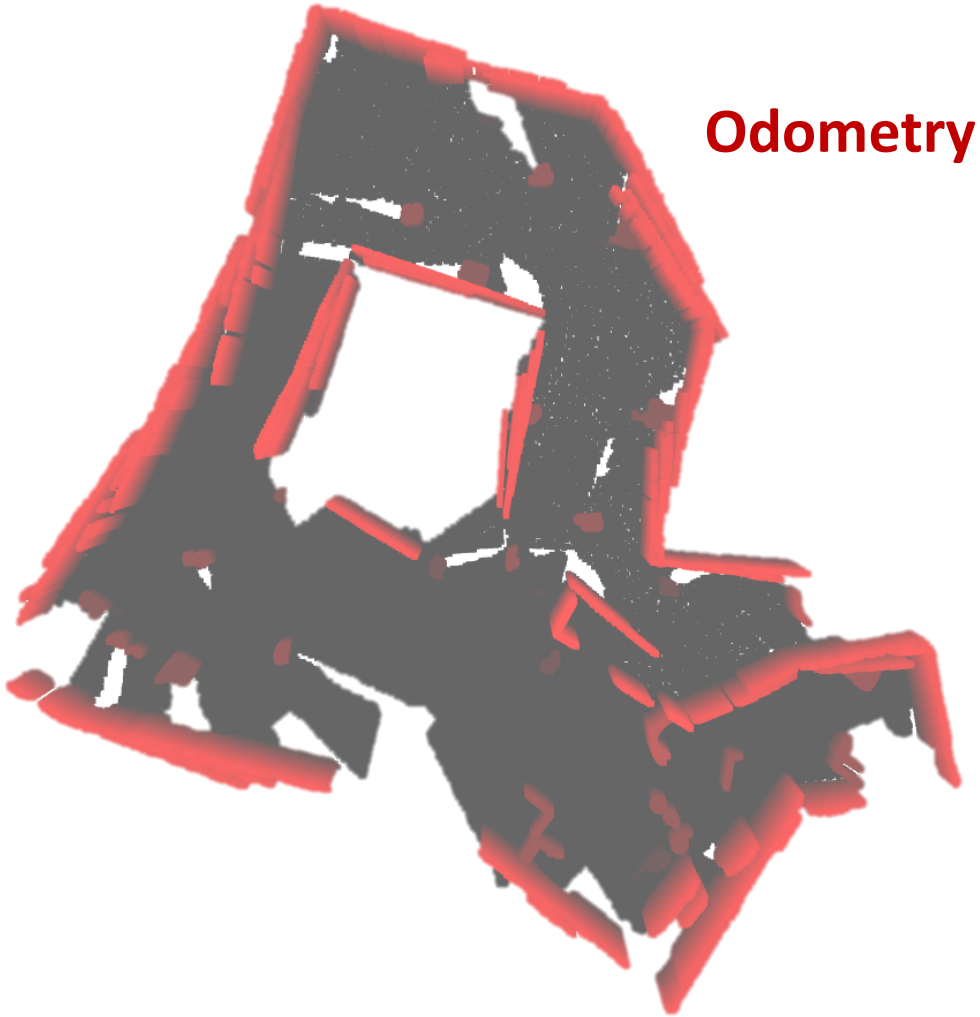




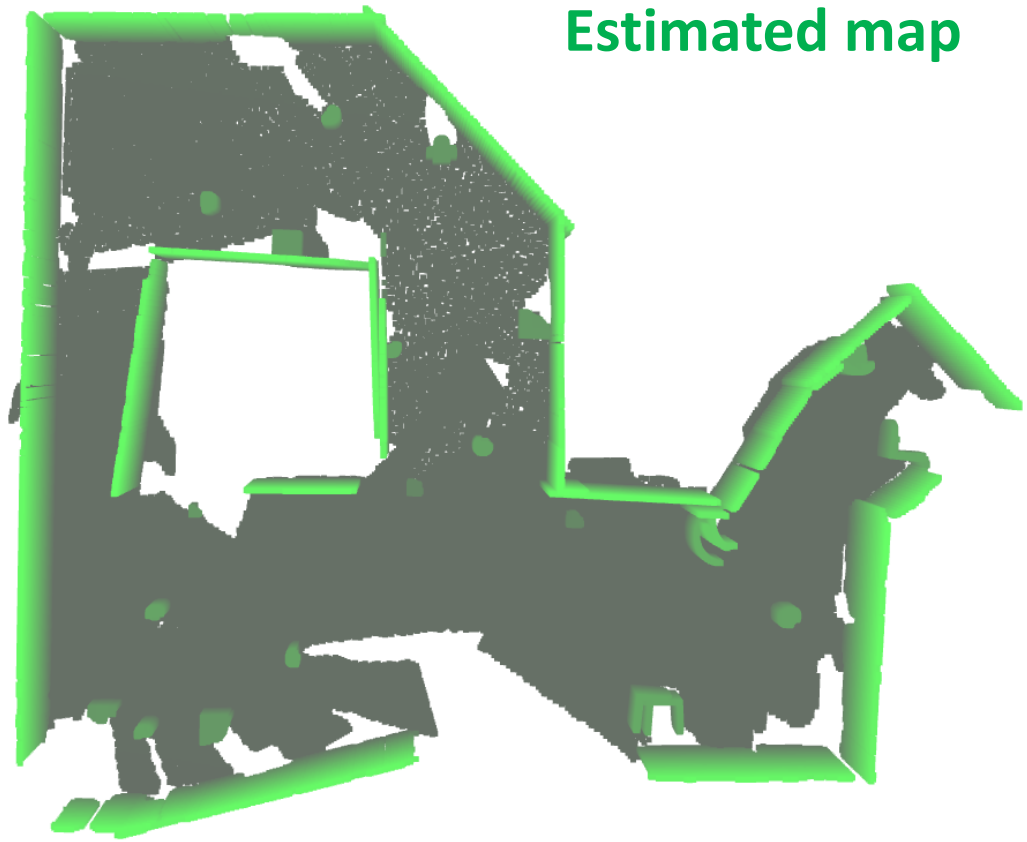


Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$



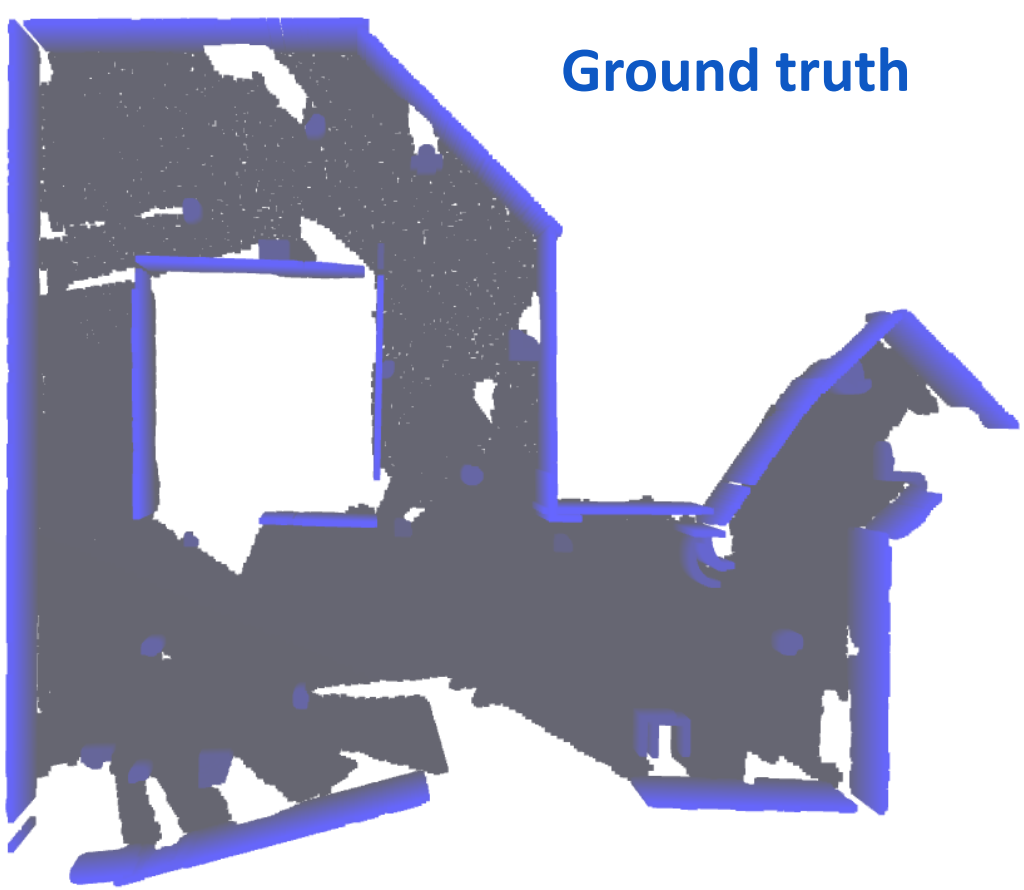
Odometry map

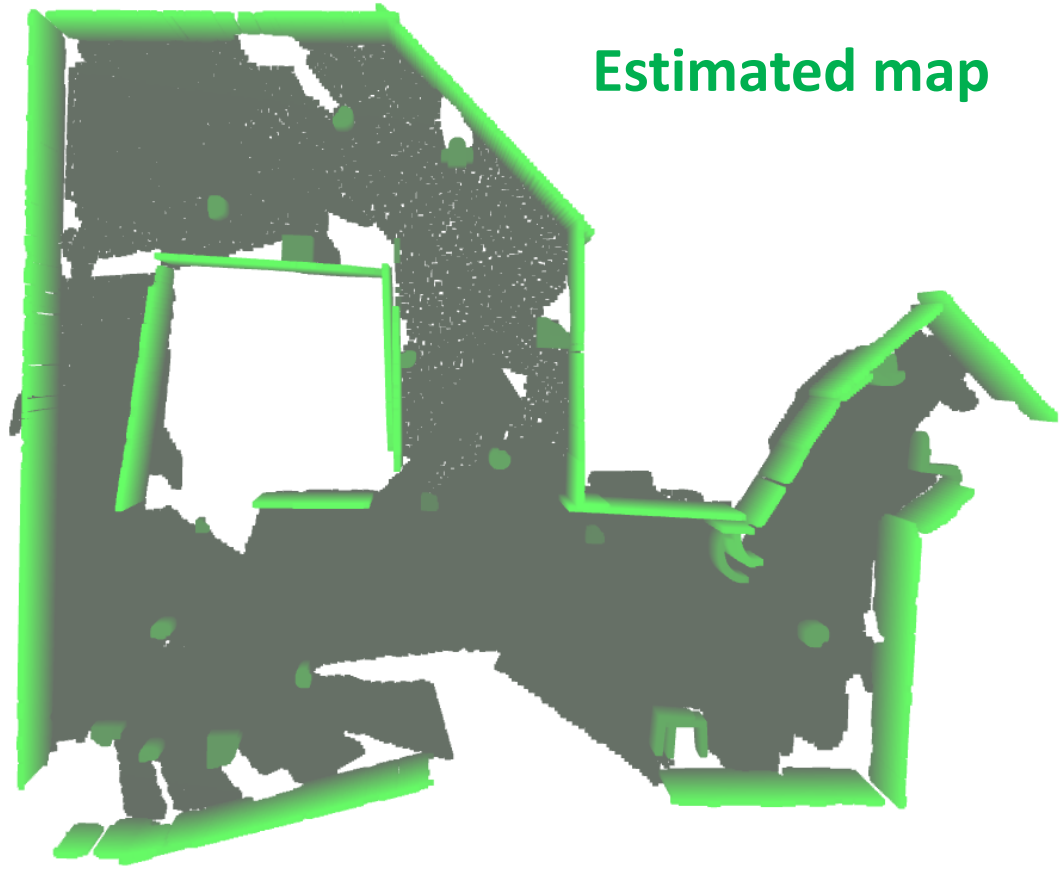


Estimated map

Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

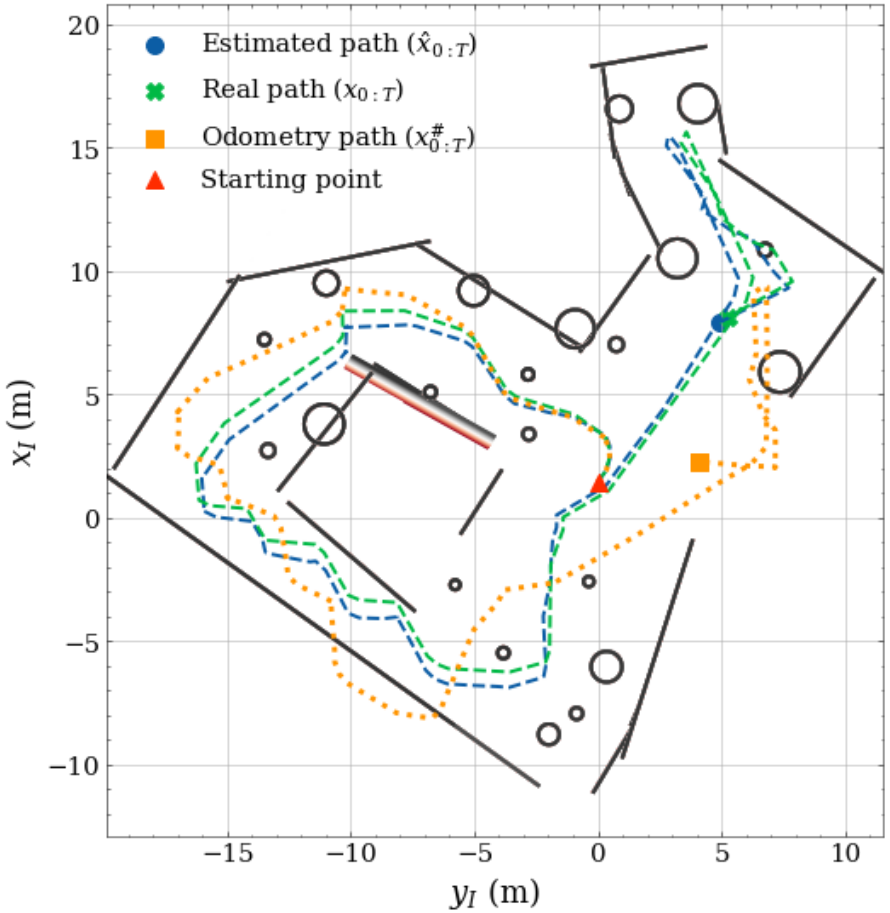




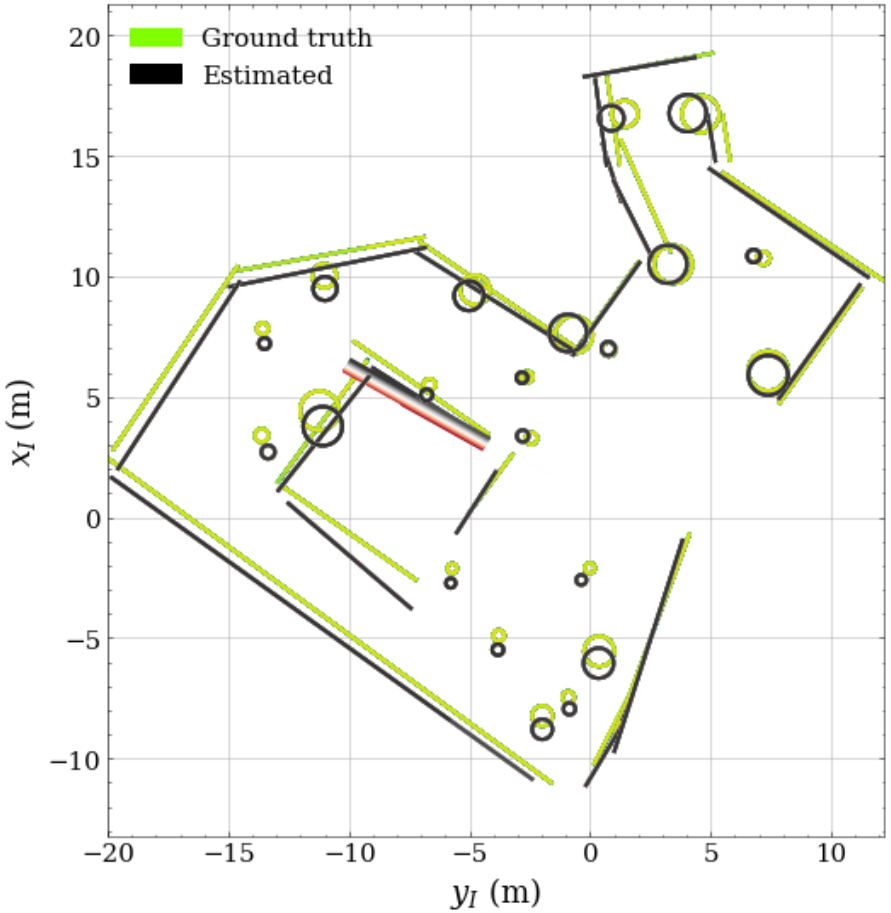
Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

Path projection



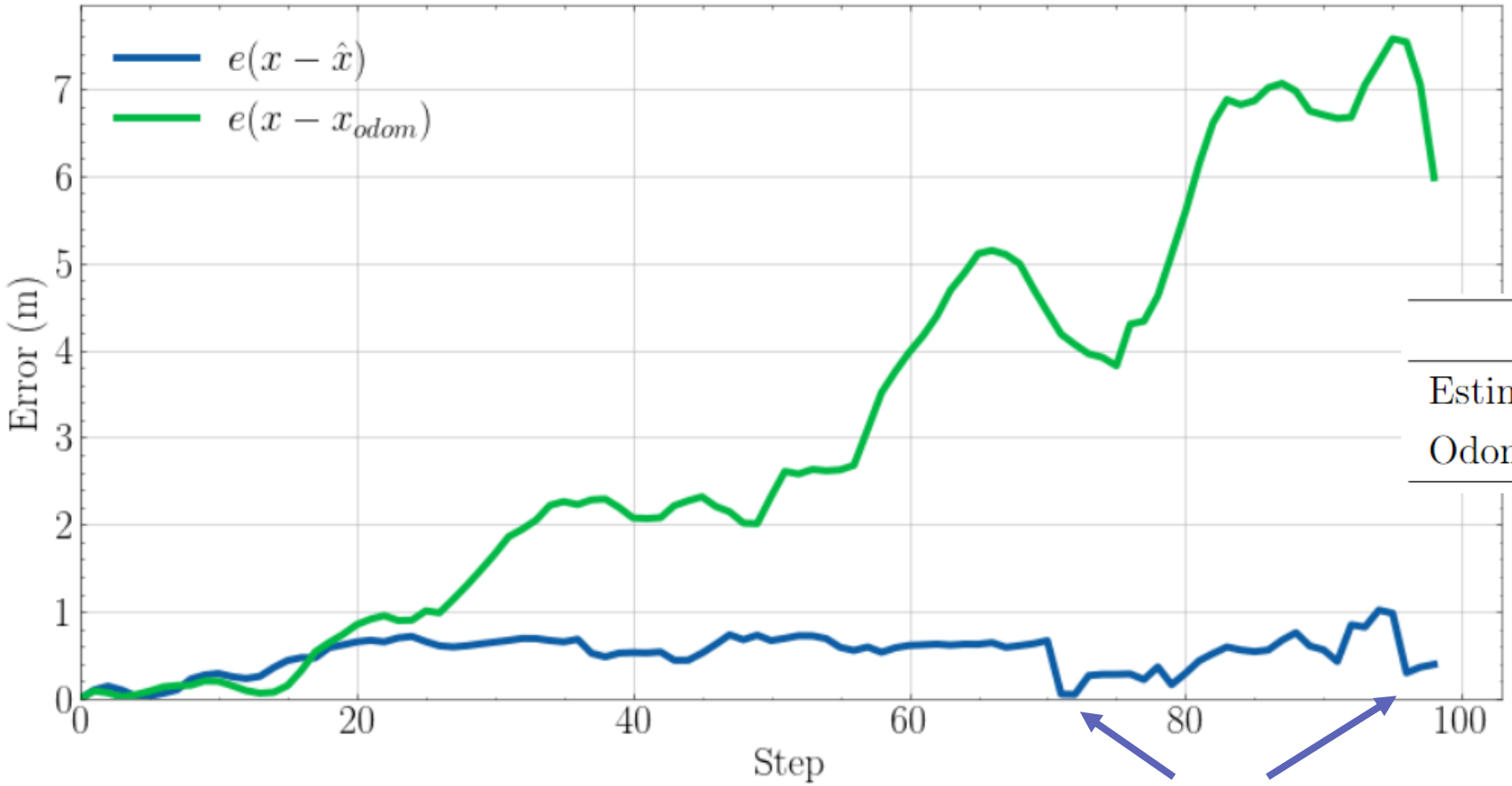
Feature projection



Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

Path Error



	IAE criteria
Estimated path error	**49.82**
Odometry path error	**304.28**

Error reduction 83,6%

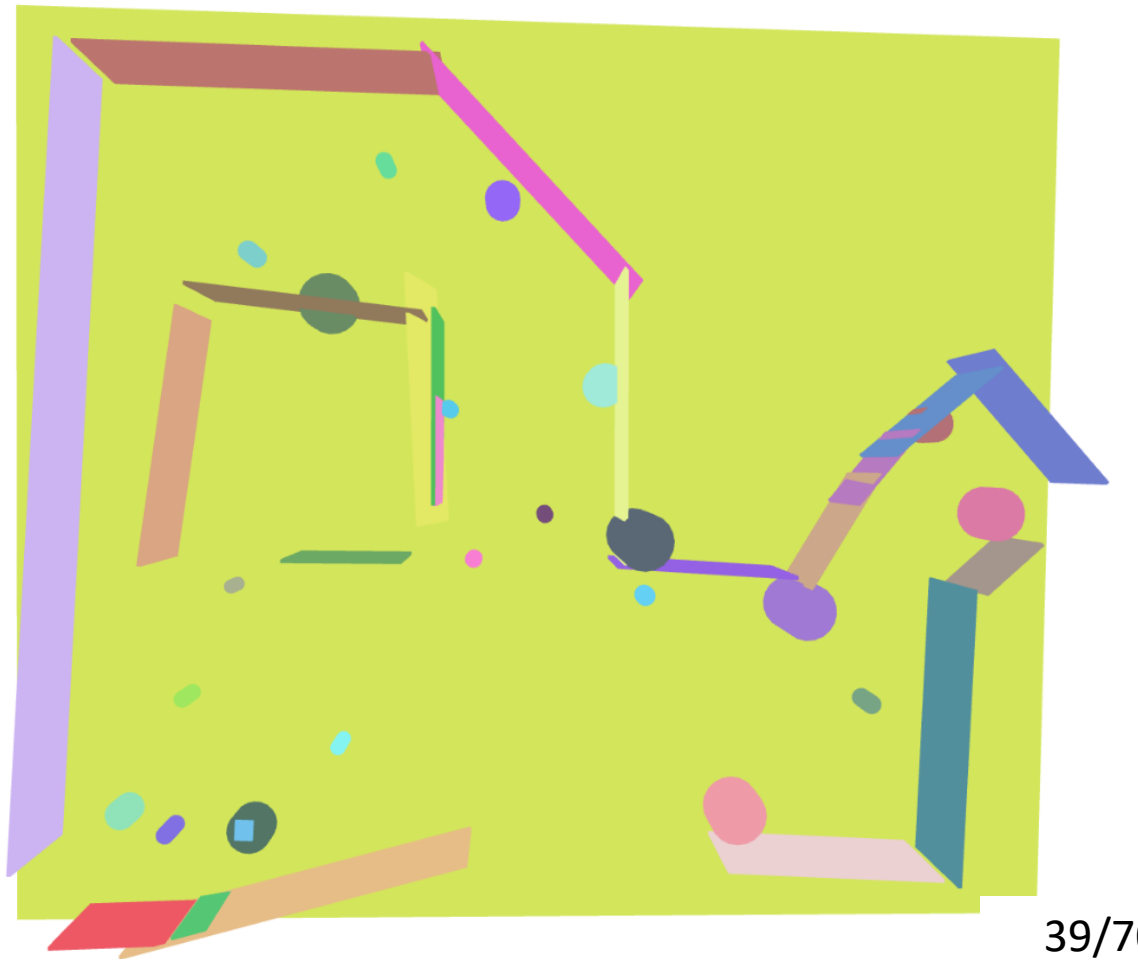
Loop closure

Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

Point cloud map

High level map

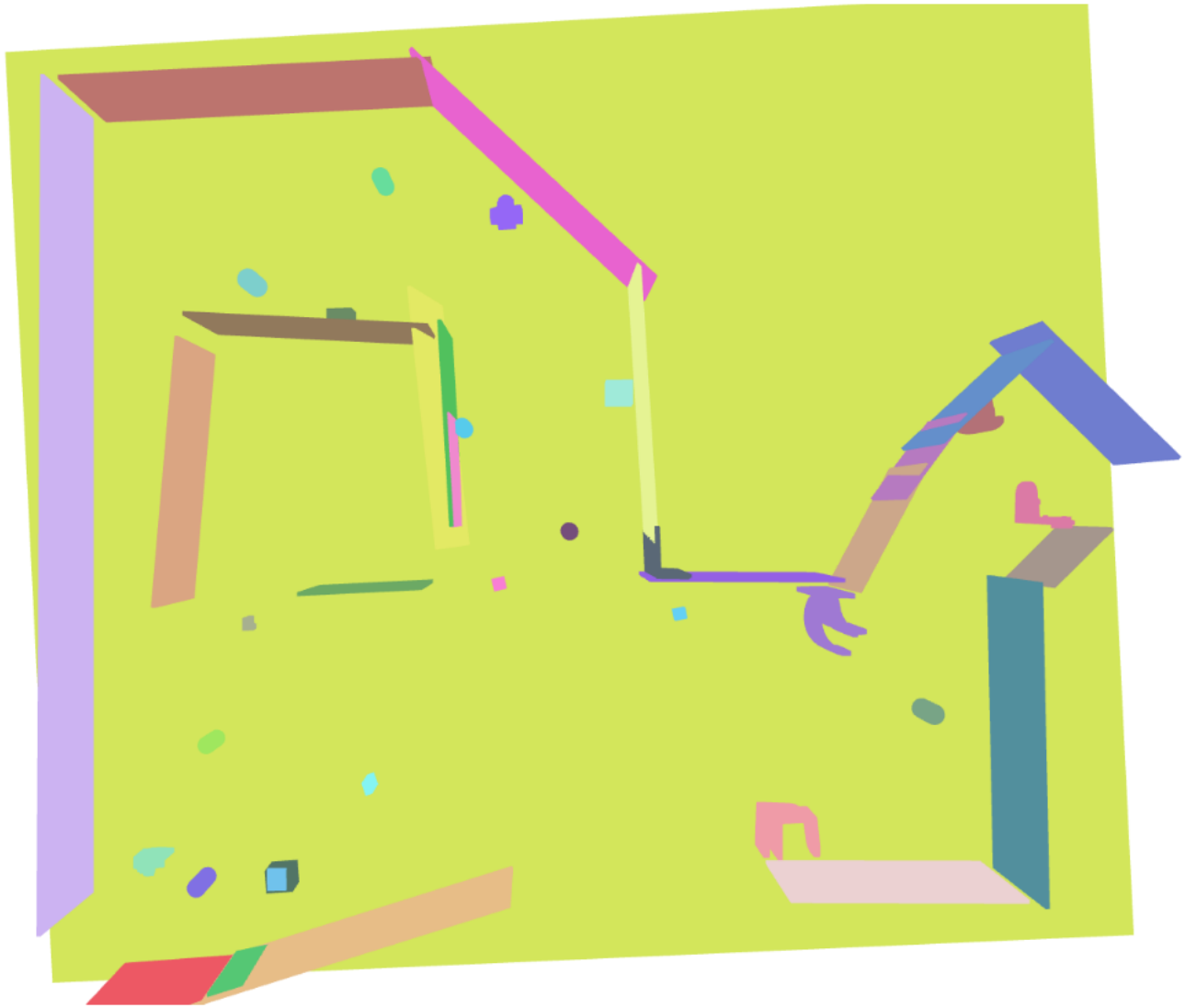


Simulation

Normal process noise $3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

Hybrid map

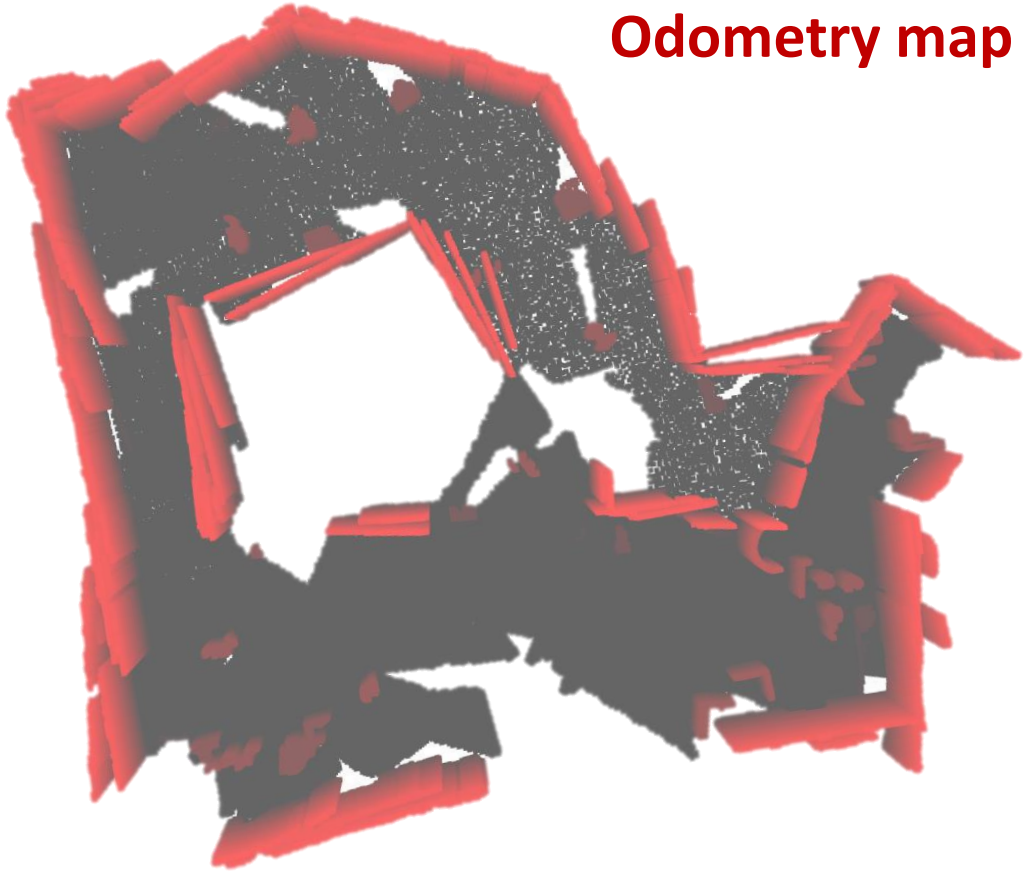
13 parameterized features
7 features as point clouds



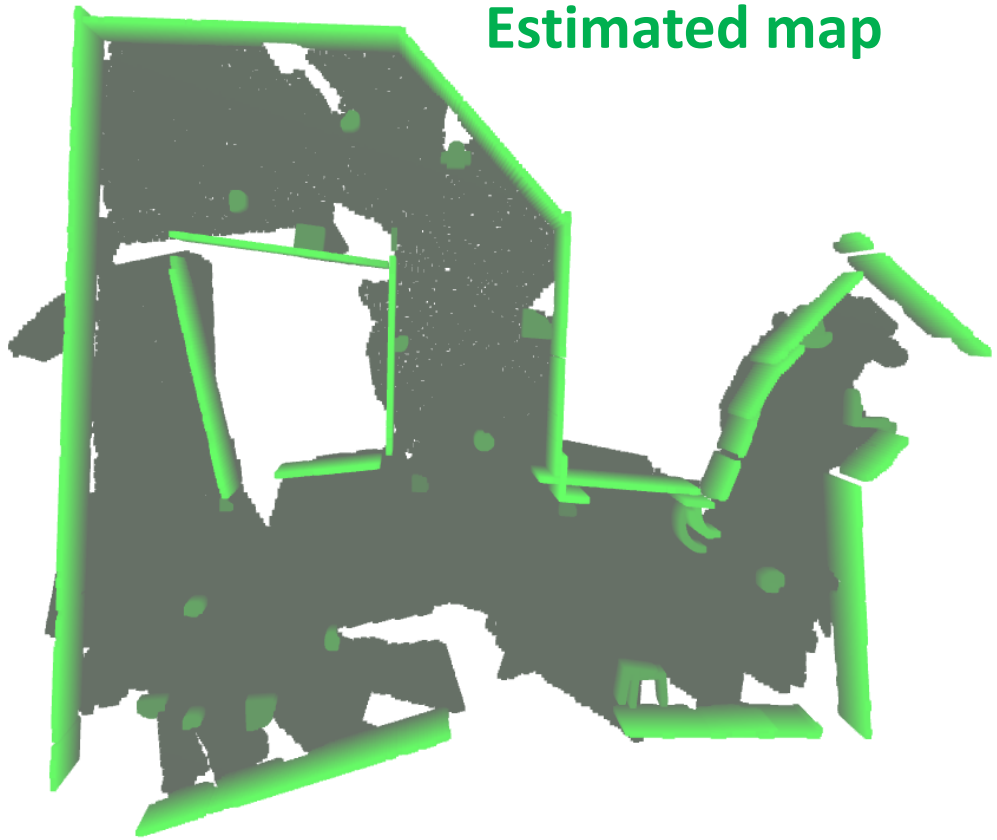
Simulation

Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

Odometry map



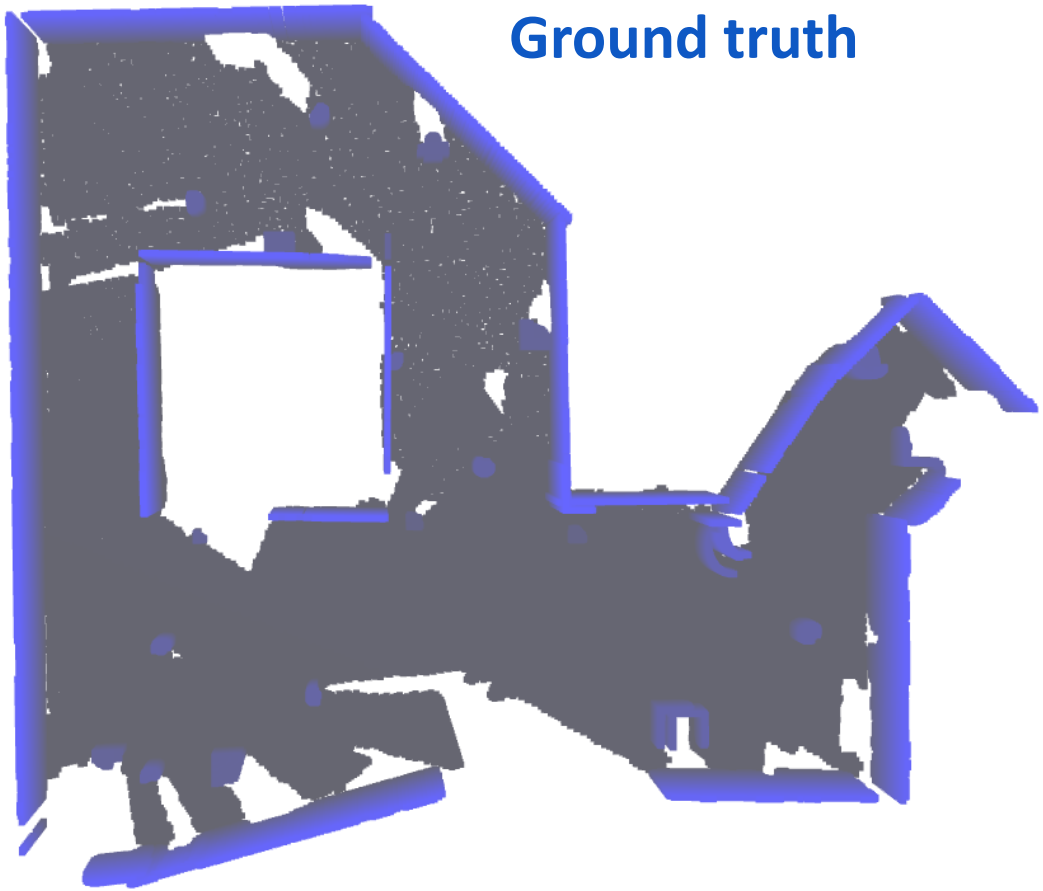
Estimated map



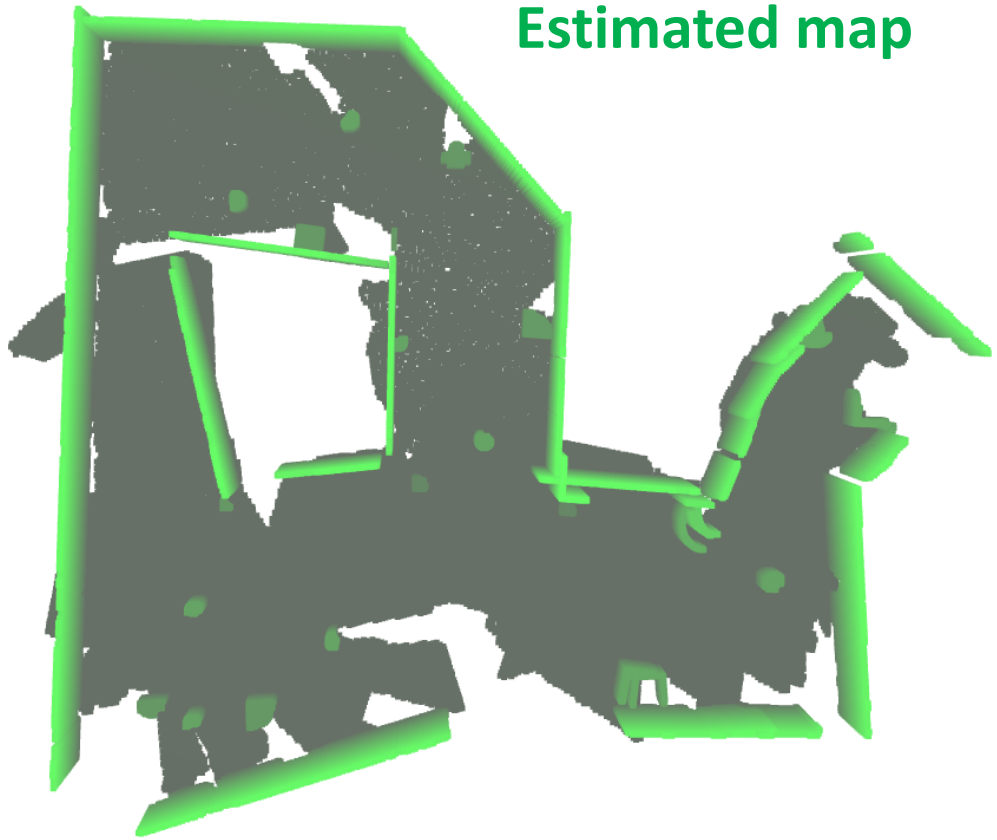
Simulation

Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

Ground truth



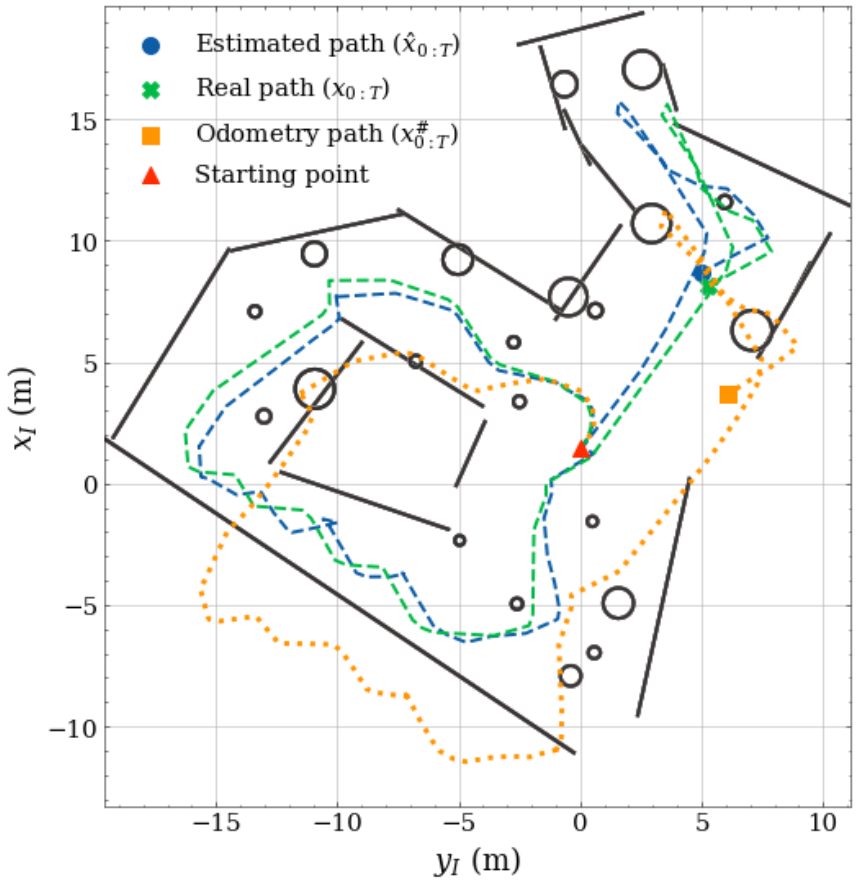
Estimated map



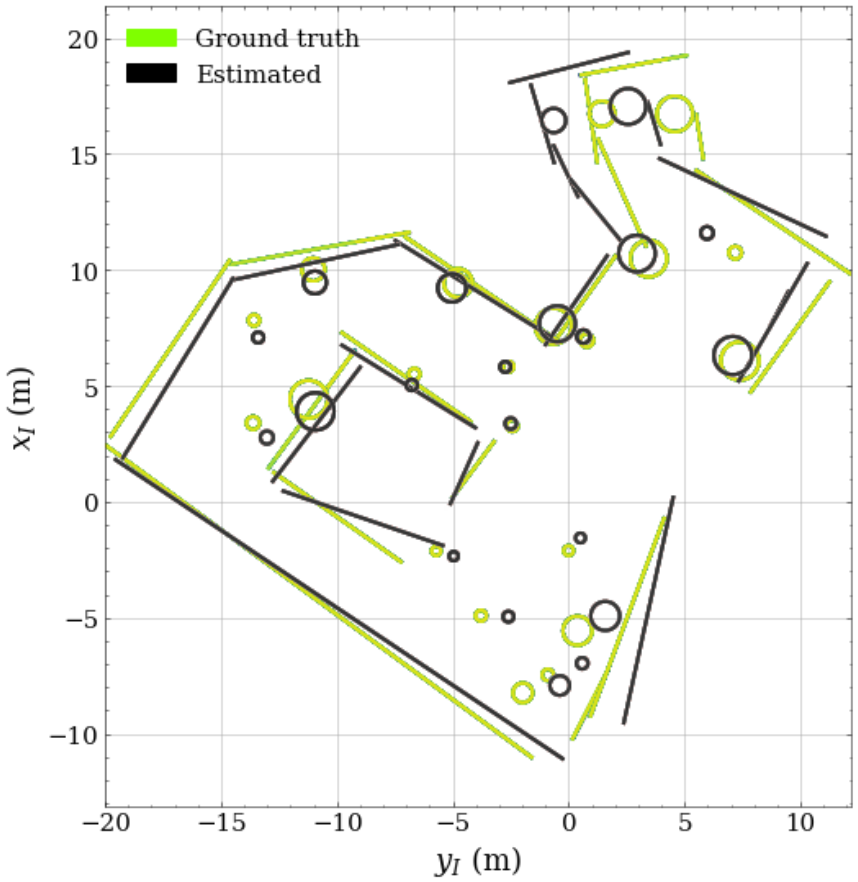
Simulation

Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

Path projection



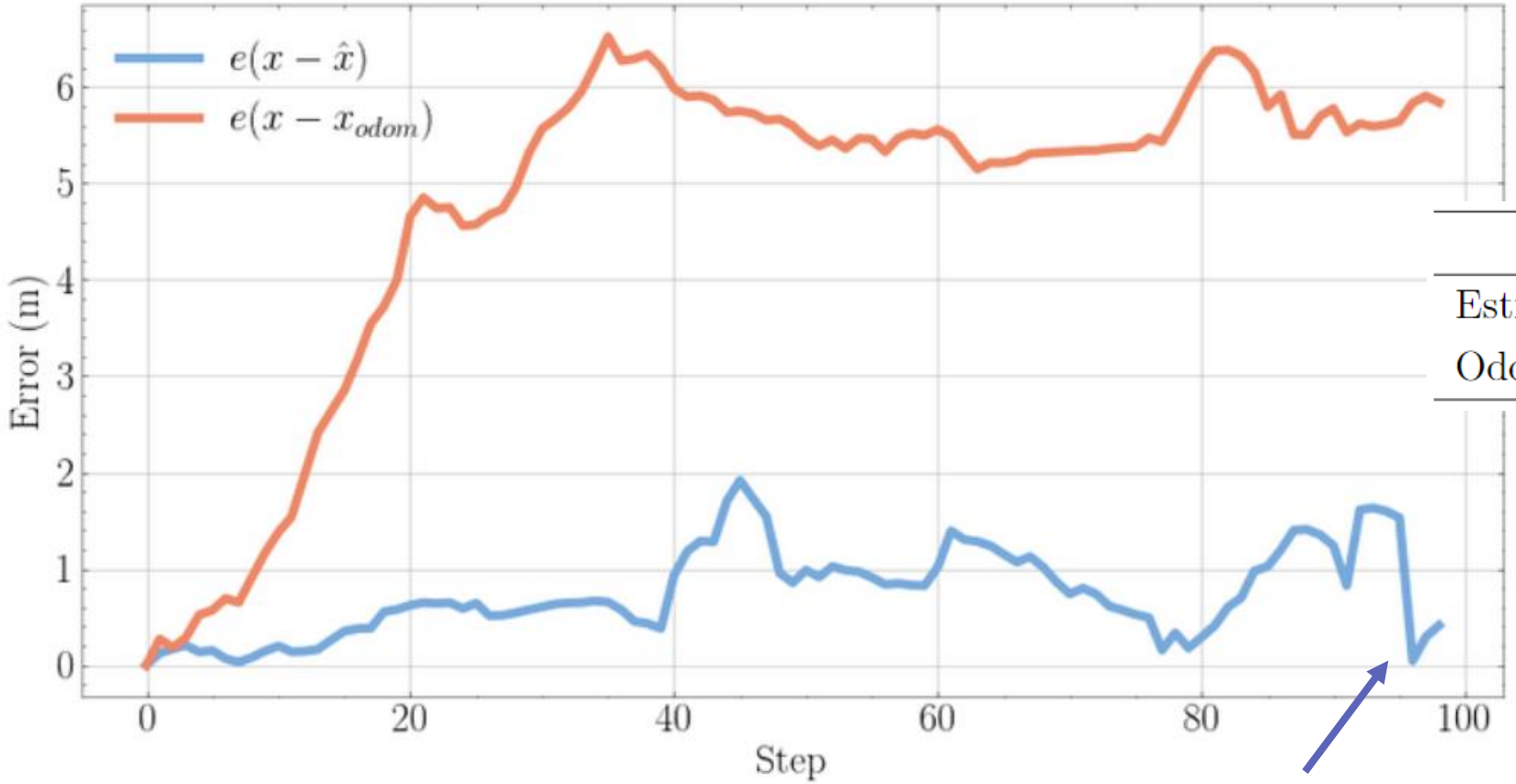
Feature projection



Simulation

Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

Path Error



	IAE criteria
Estimated path error	**73.5**
Odometry path error	**472.3**

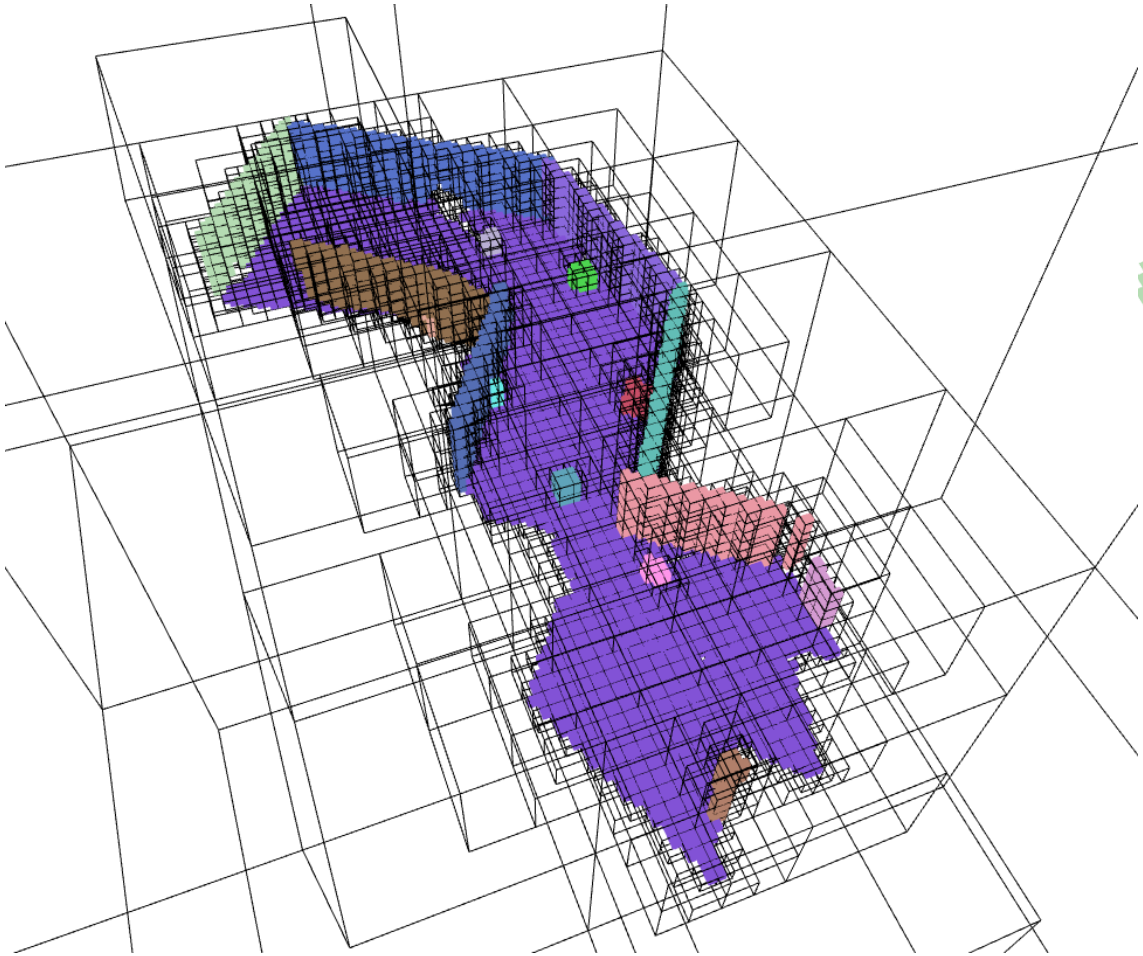
Error reduction 84,4%

Loop closure

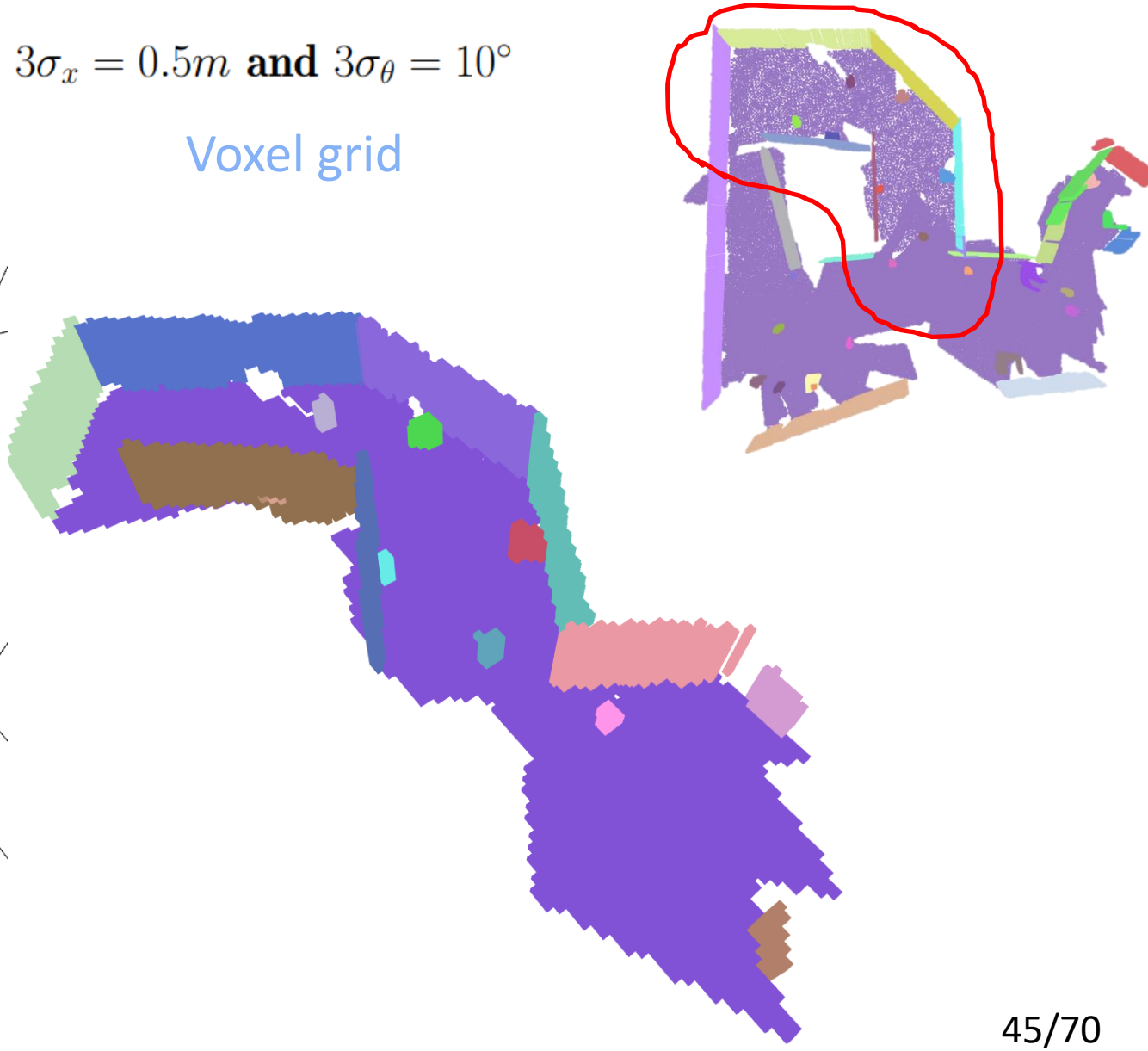
Simulation

Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

Octree

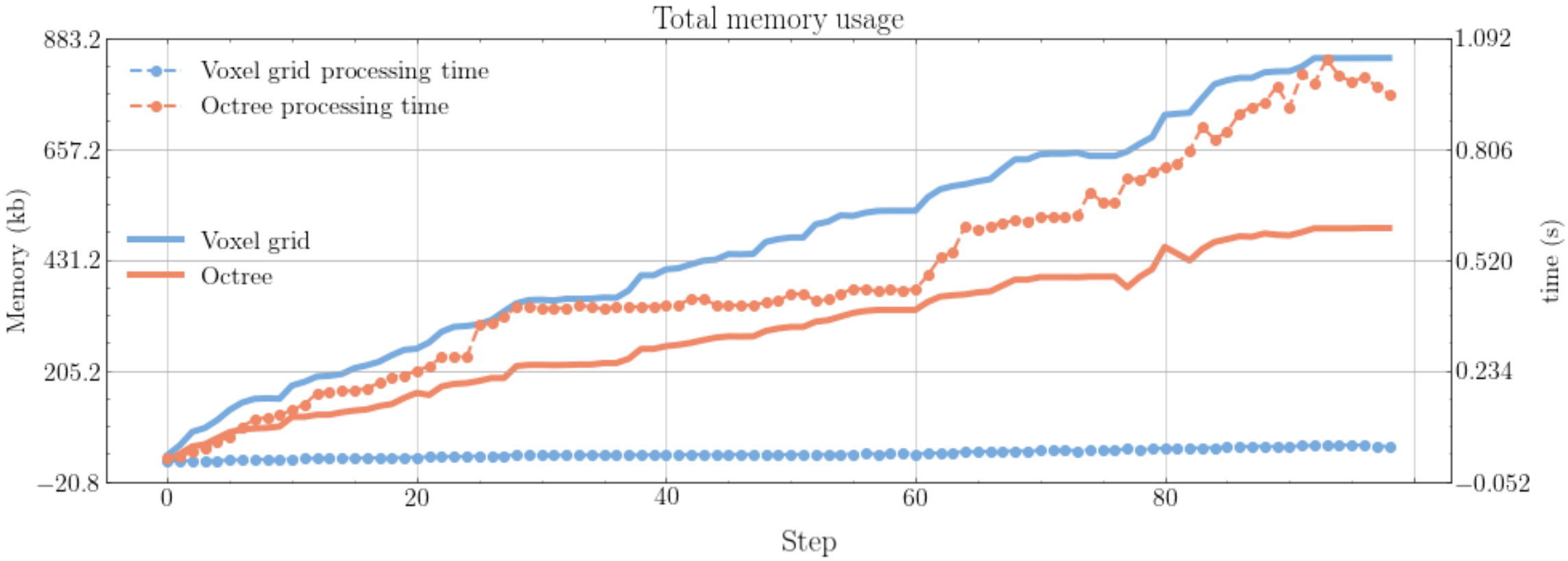


Voxel grid



Simulation

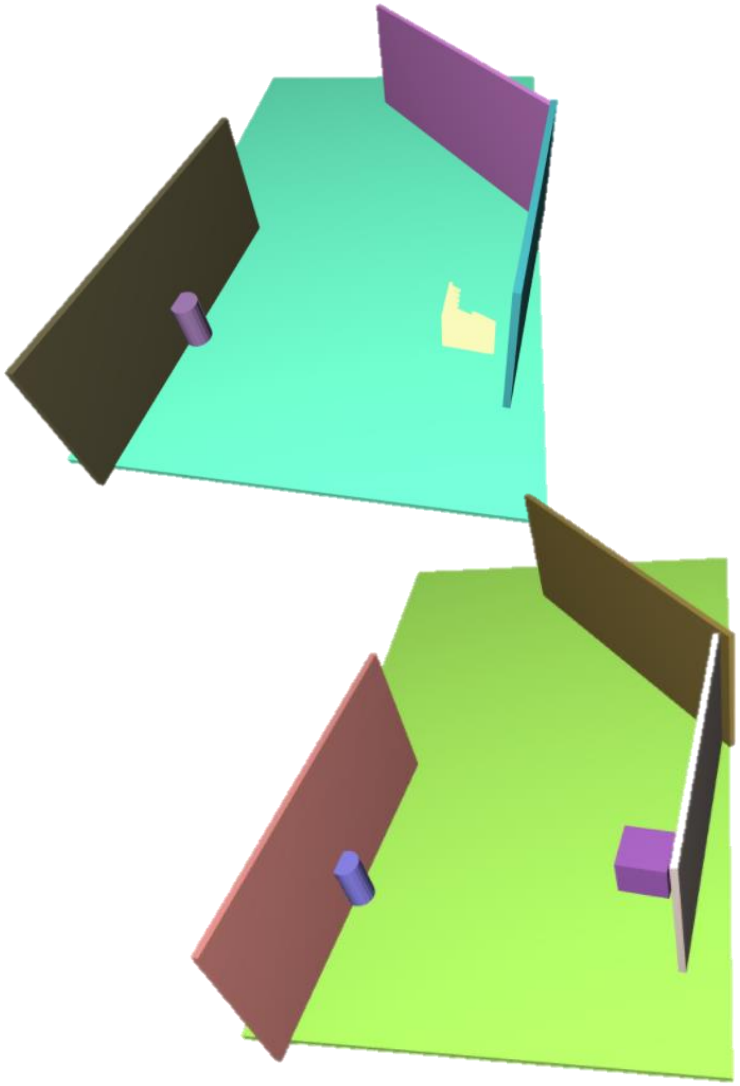
Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

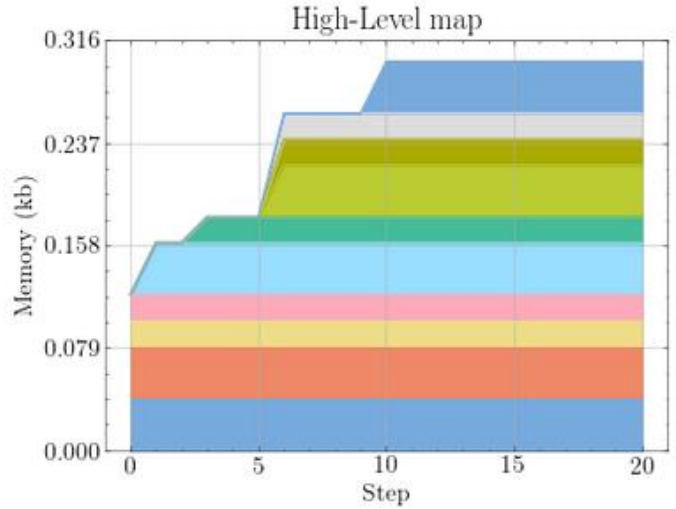


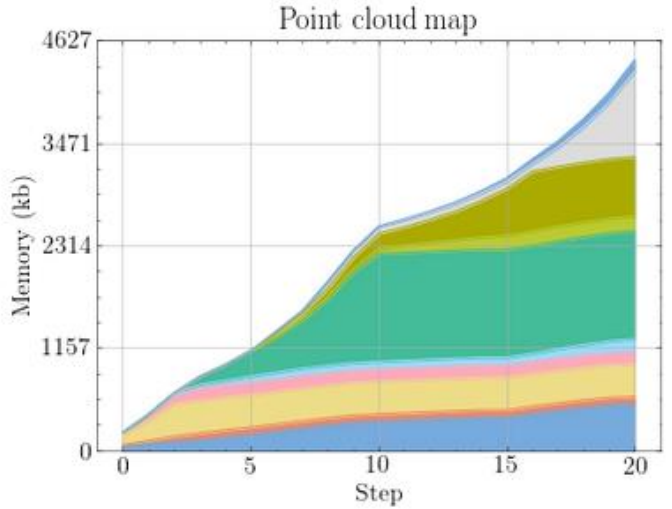
Simulation

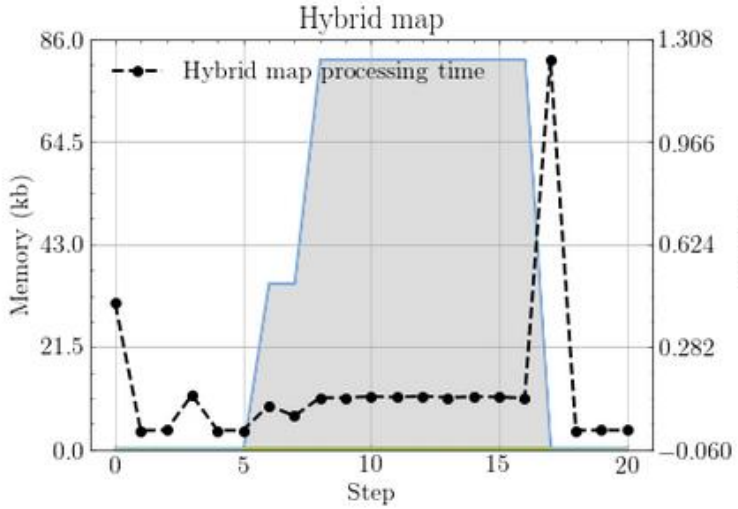
Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

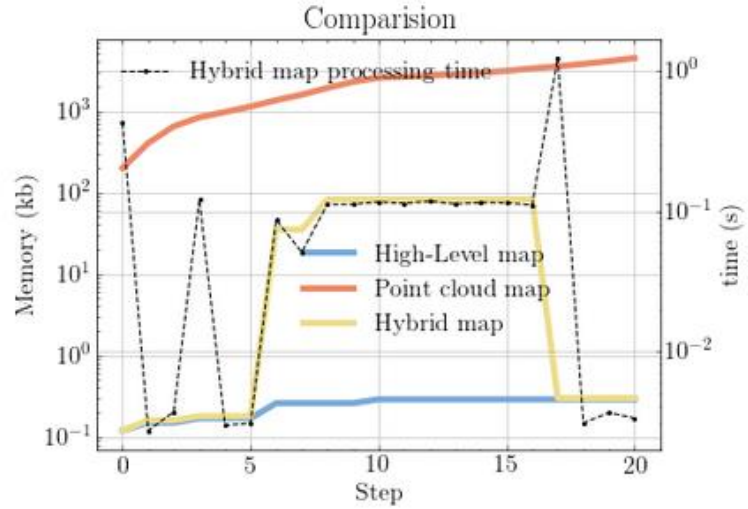
Color: Memory usage per feature









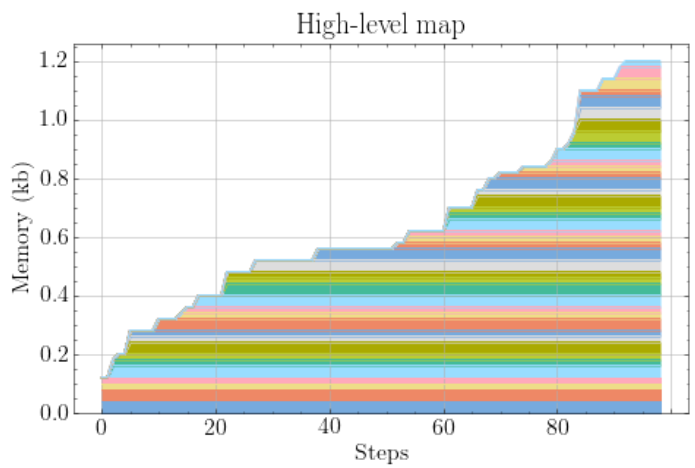


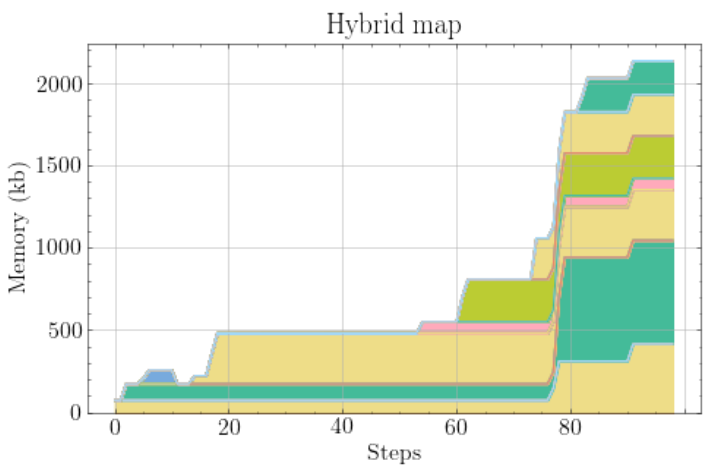
Simulation

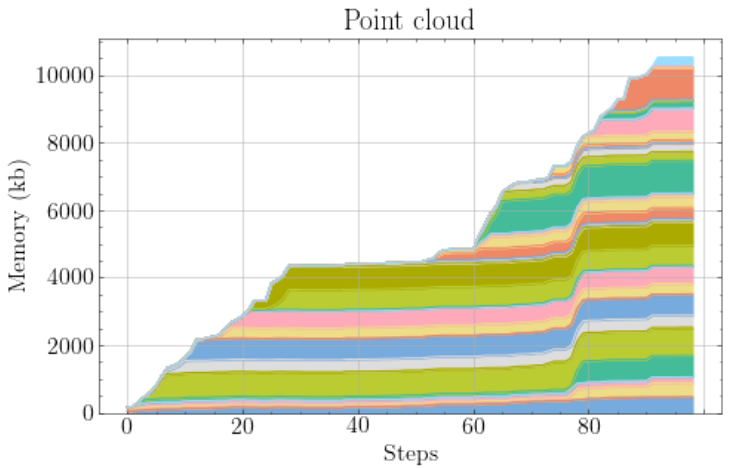
Increased process noise $3\sigma_x = 0.5m$ **and** $3\sigma_\theta = 10°$

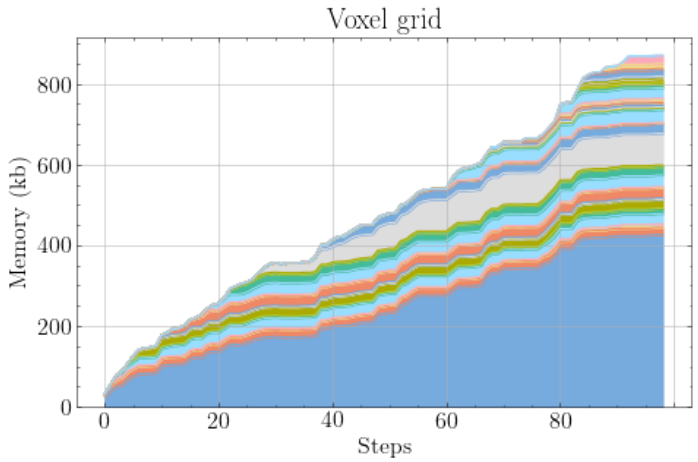
Color: Memory usage per feature

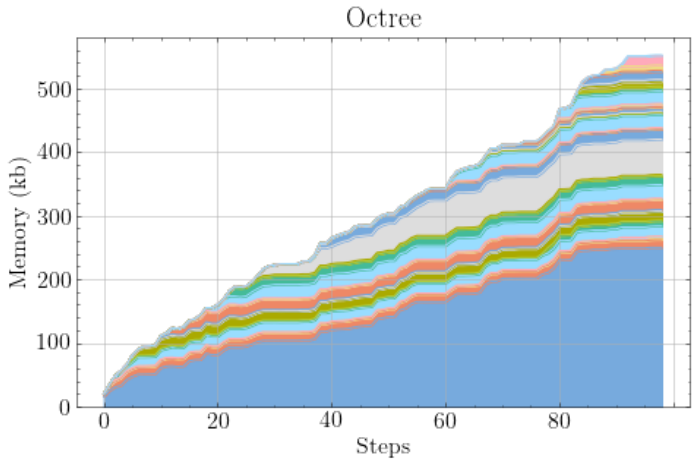
Memory usage

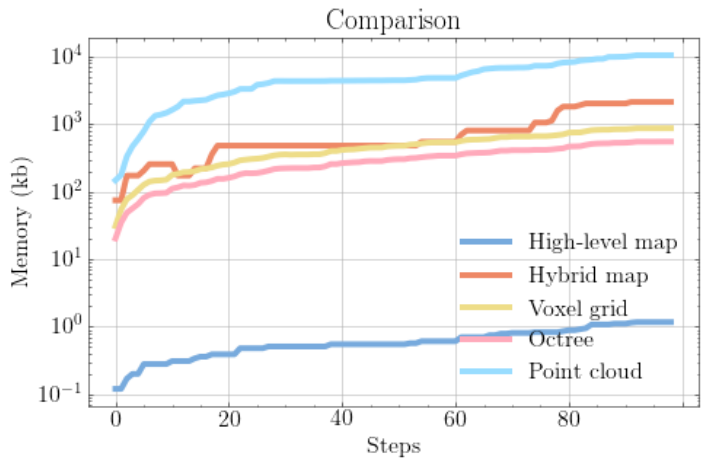












Experimental results

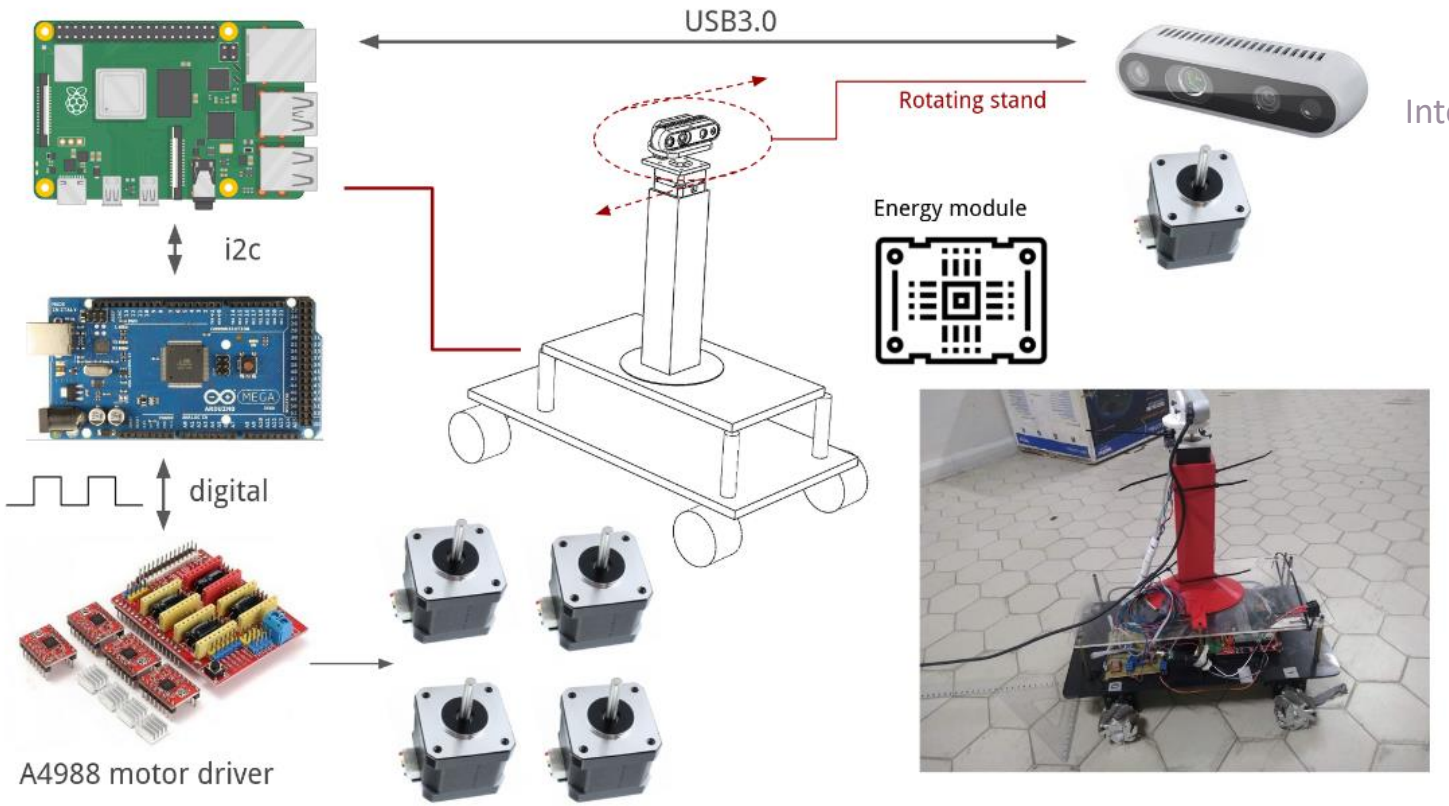
Setup

	Specification
RGBD camera	**Intel Realsense D435i** Depth range: 0.3 to 3 m Depth resolution: up to 1280 x 720 Depth accuracy: $< 2\%$ at 2 m
Processing board	Raspberry Pi 4B Arduino Mega 250
Motor driver	4x A4988 (+1 for camera rotation)
Motor type	4x Stepper motor (+1 for camera rotation)

Omni robot

Raspberry pi 4B

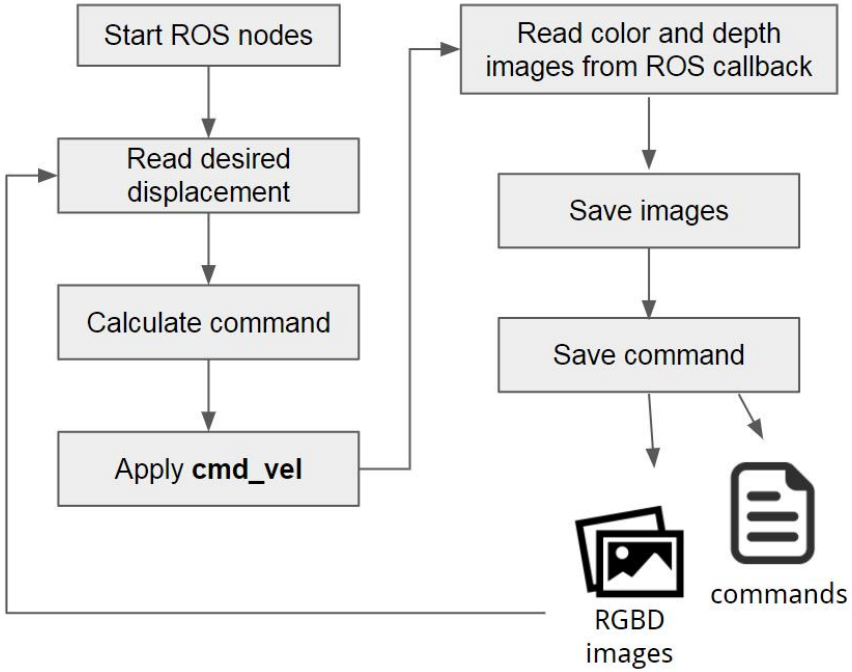
Arduino Mega 250



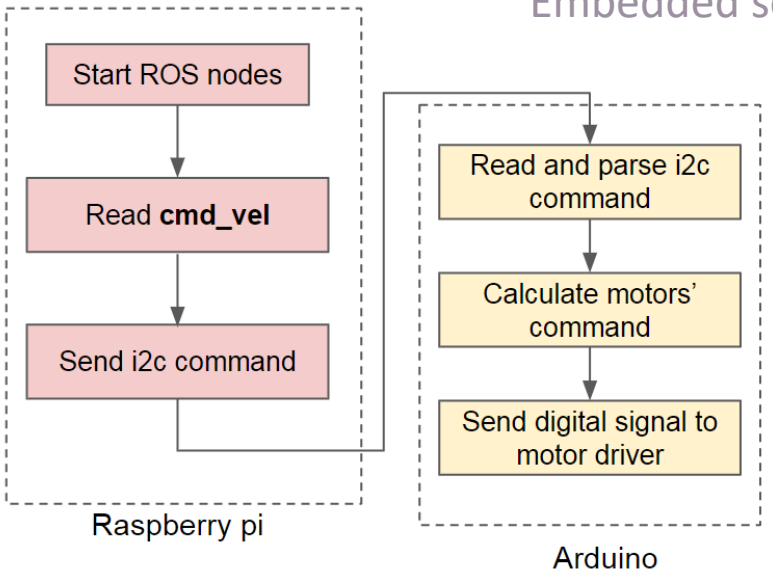
Intel RealSense D435i

Experimental results

Setup



Embedded software



LMI corridor

Setup

- Desired trajectory

Command $u_k = [\delta_x \quad \delta_\theta]^T$

$$\hat{V} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

$3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

$$\hat{W} = \begin{bmatrix} \sigma_X^2 & 0 & 0 \\ 0 & \sigma_Y^2 & 0 \\ 0 & 0 & \sigma_Z^2 \end{bmatrix} \quad \sigma_X = \sigma_Y = \sigma_Z = \frac{0.1}{3}$$

LMI corridor

Setup

- Desired trajectory

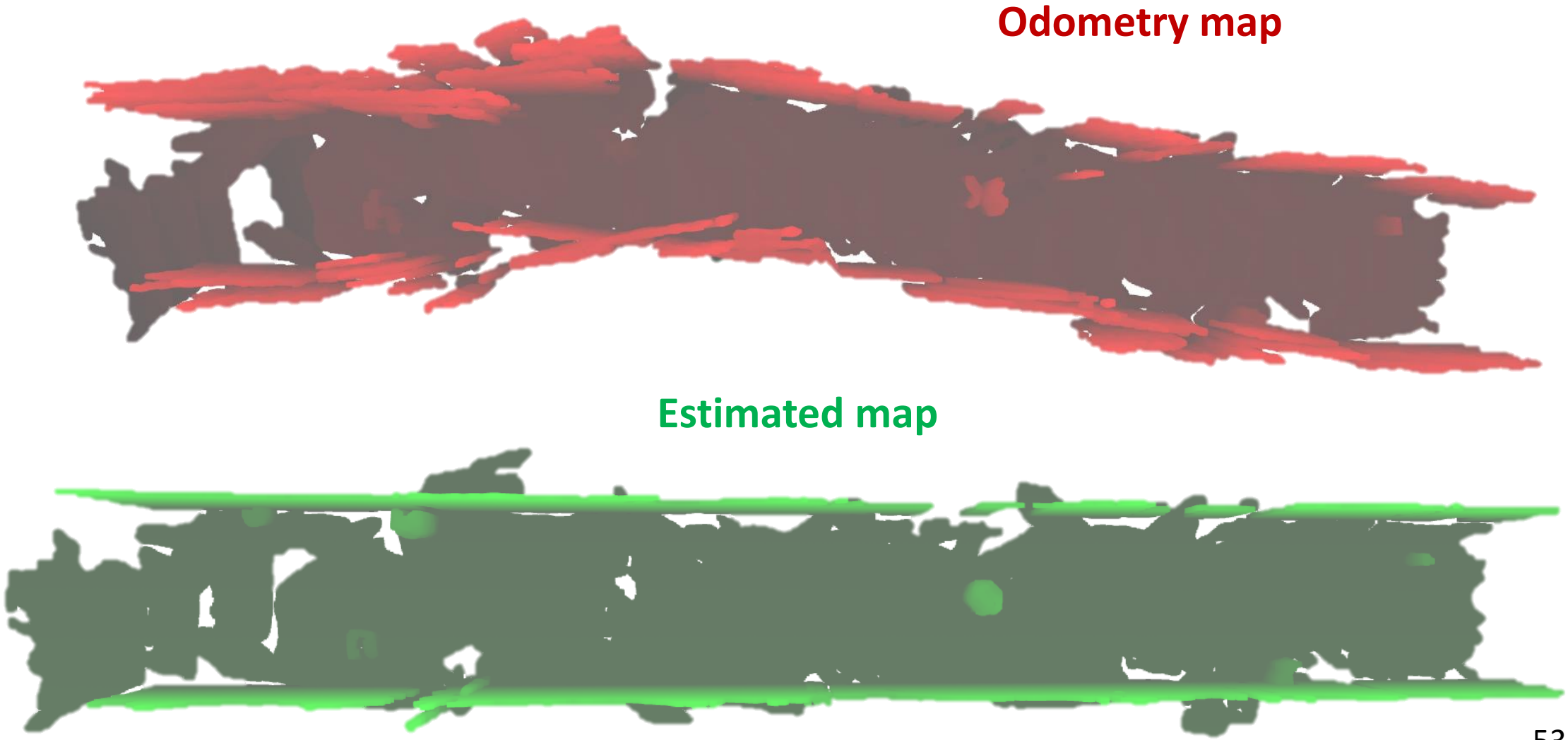
Command $u_k = [\delta_x \quad \delta_\theta]^T$

$$\hat{V} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

$3\sigma_x = 0.2m$ **and** $3\sigma_\theta = 5°$

$$\hat{W} = \begin{bmatrix} \sigma_X^2 & 0 & 0 \\ 0 & \sigma_Y^2 & 0 \\ 0 & 0 & \sigma_Z^2 \end{bmatrix} \quad \sigma_X = \sigma_Y = \sigma_Z = \frac{0.1}{3}$$

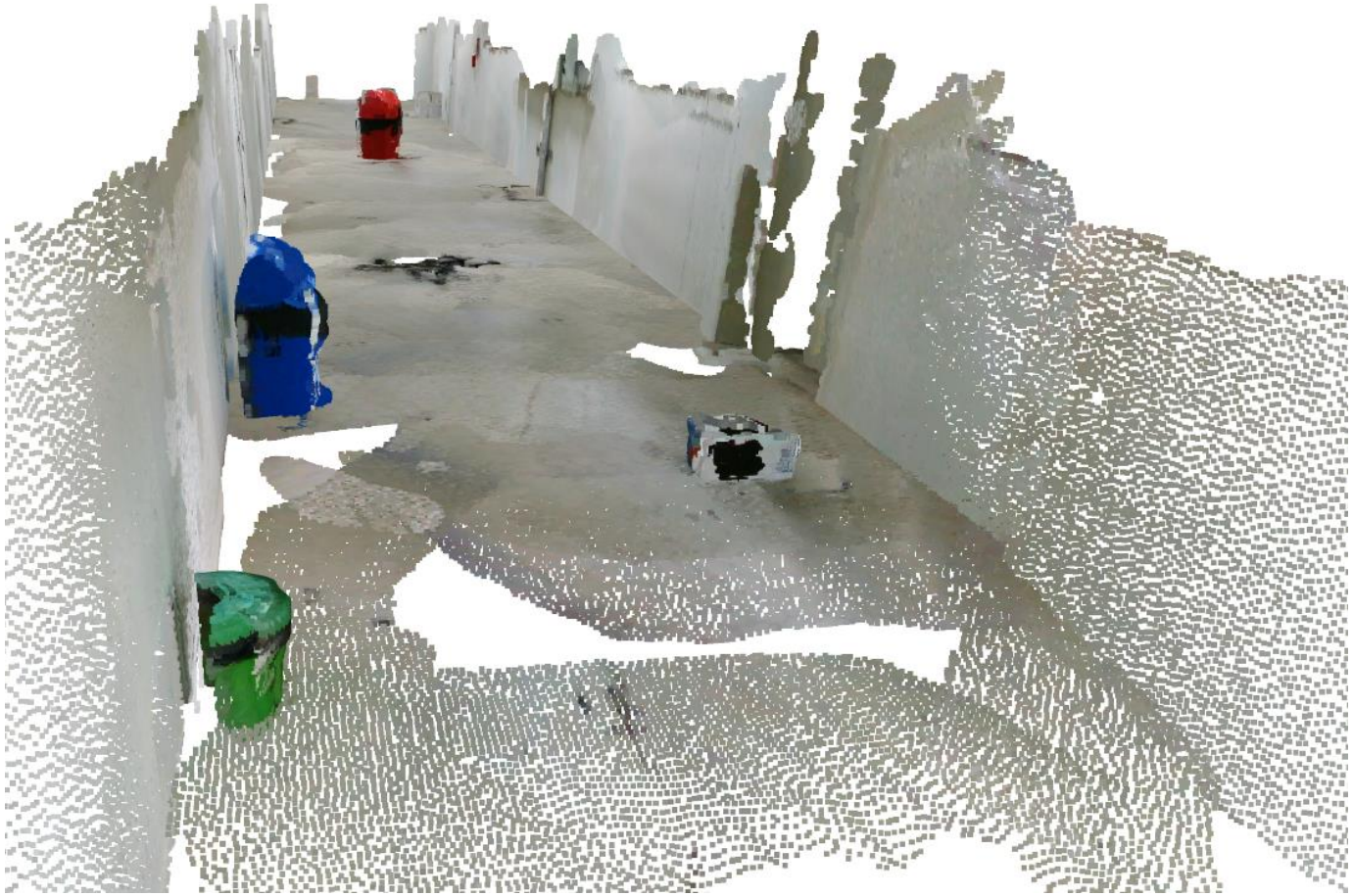
LMI corridor Result



LMI corridor Result

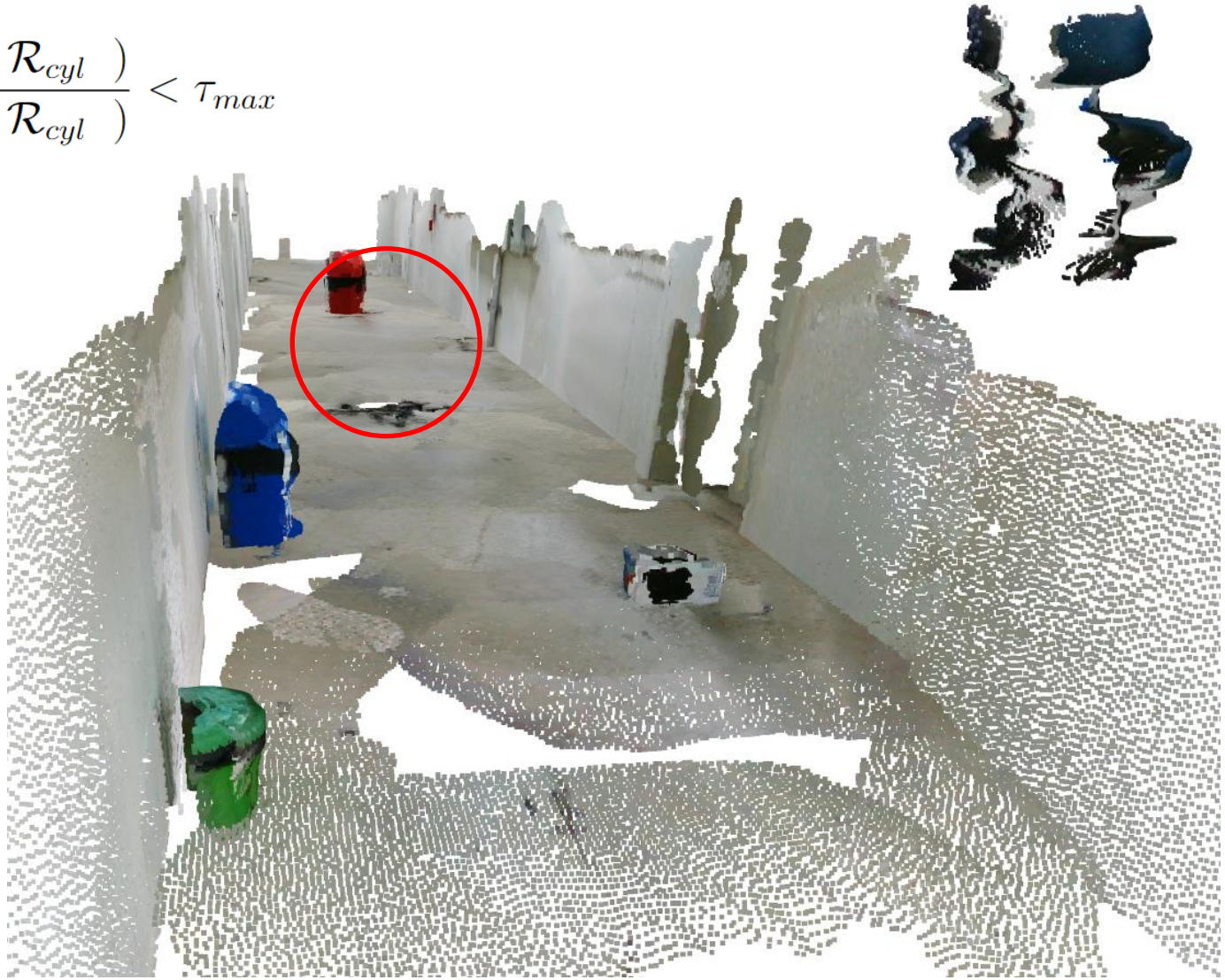
Photo

Point cloud map



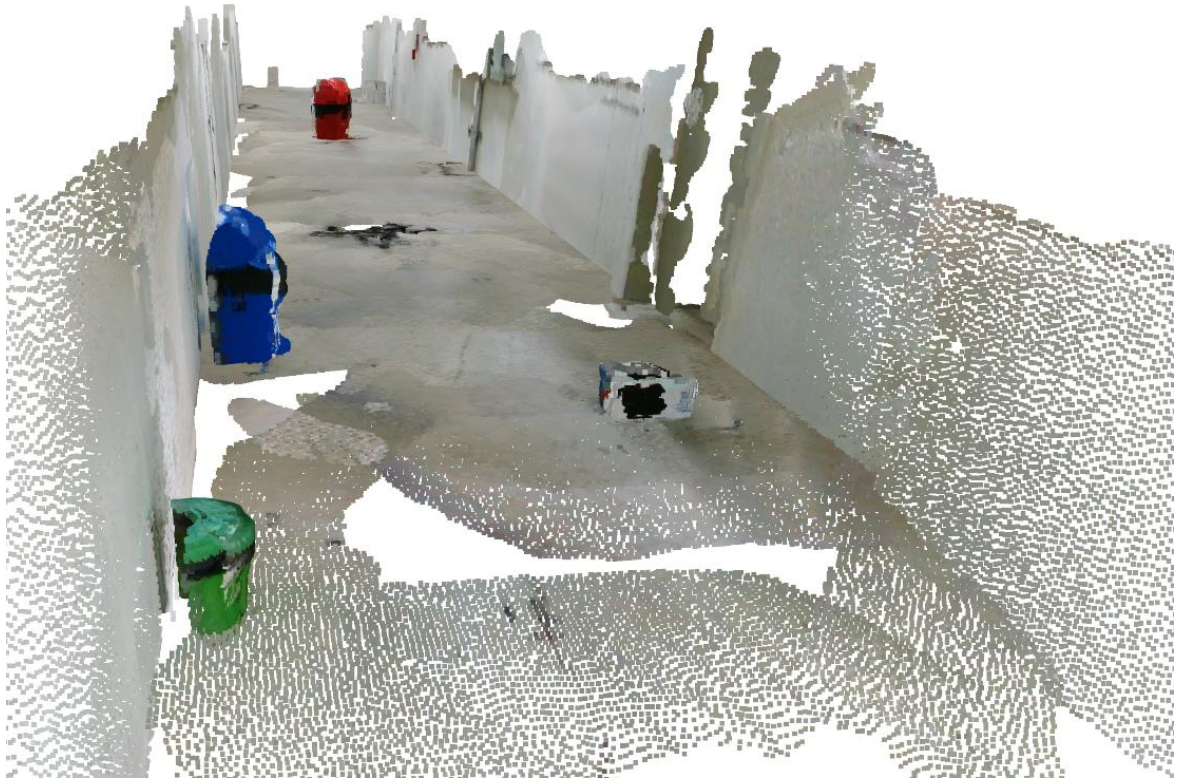
LMI corridor Result

$$\frac{\sigma(\ \mathcal{R}_{cyl}\ )}{\mu(\ \mathcal{R}_{cyl}\ )} < \tau_{max}$$

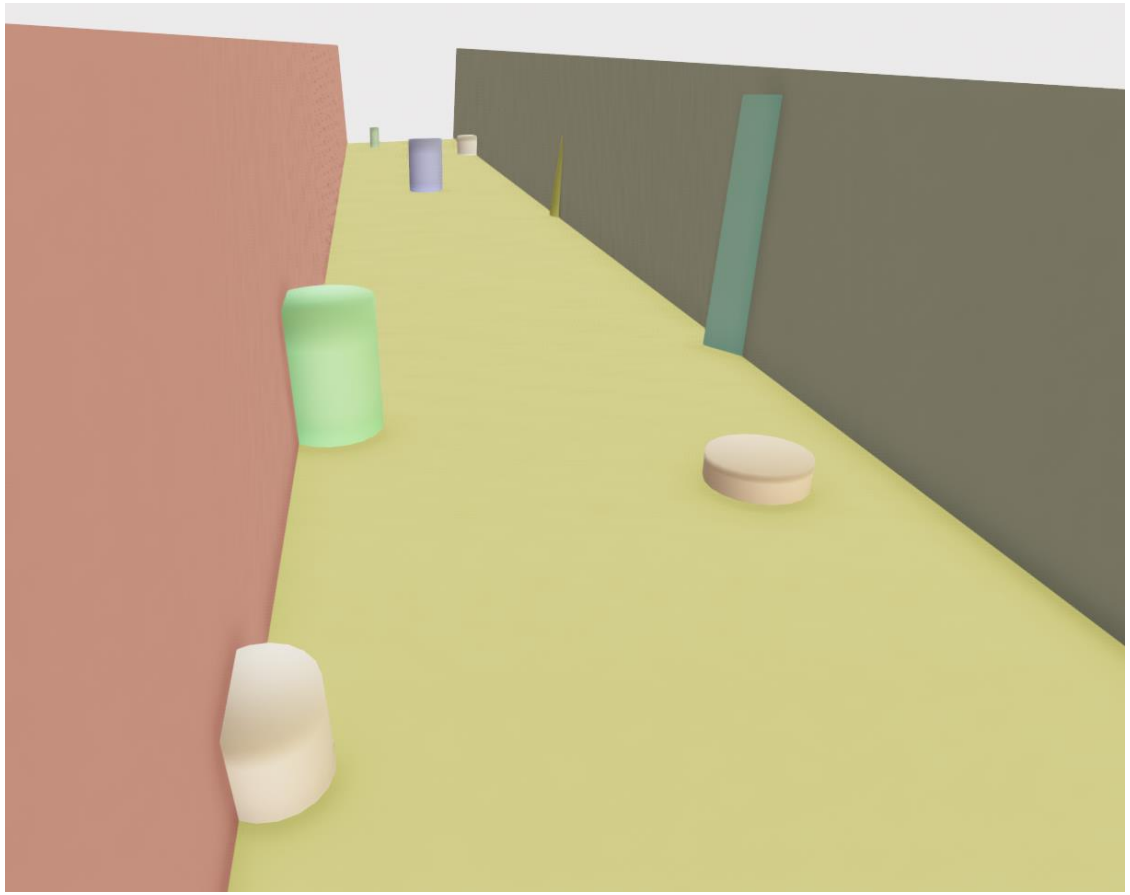


LMI corridor Result

Point cloud map

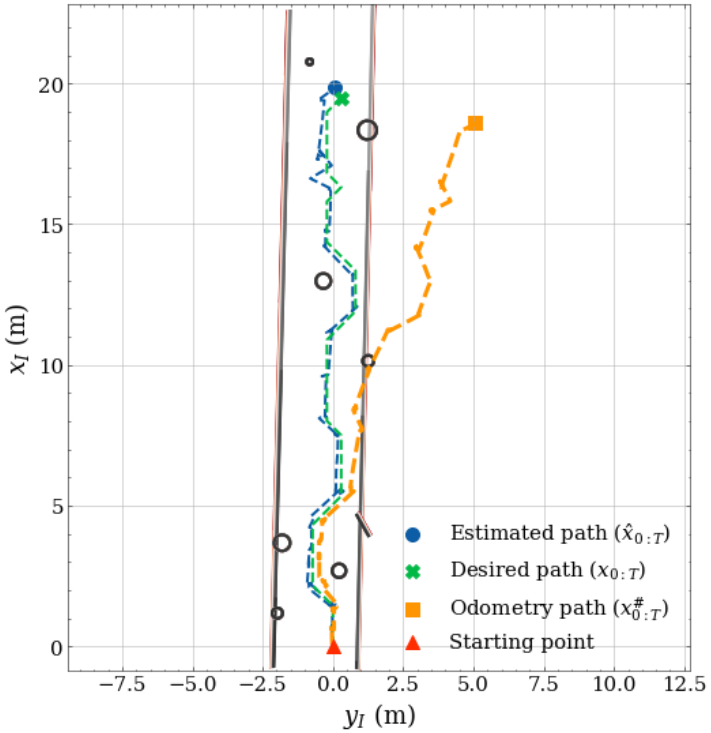


High level map



LMI corridor

Path projection



Feature projection

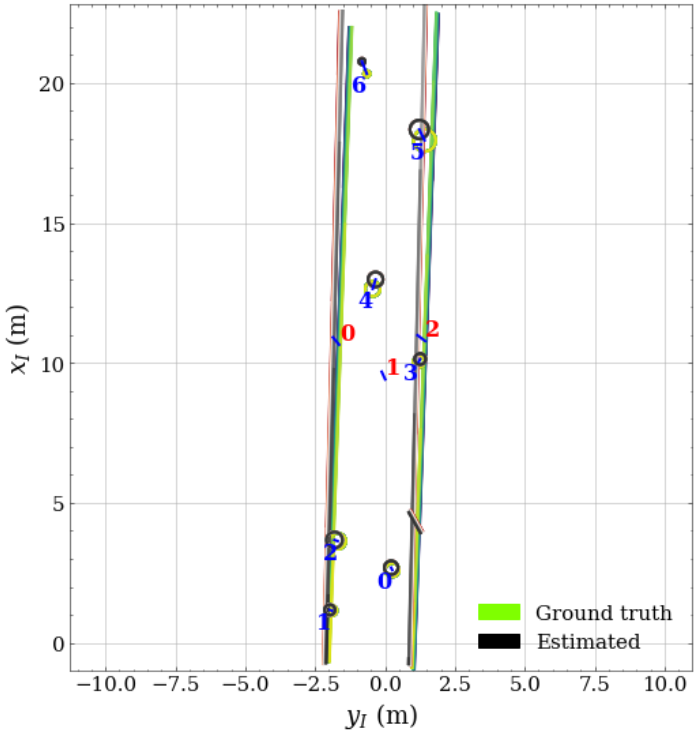


TABLE 5.3 – Cylinder position error.

	c_x	c_y	c_z	$\hat{c}_x$	$\hat{c}_y$	$\hat{c}_z$	Error
0	2.60	0.26	0.41	2.68	0.22	0.41	0.09
1	1.15	-1.90	0.37	1.18	-1.98	0.37	0.08
2	3.63	-1.69	0.17	3.67	-1.81	0.17	0.13
3	10.04	1.21	-0.00	10.14	1.26	-0.01	0.11
4	12.65	-0.44	0.12	12.98	-0.34	0.12	0.35
5	17.95	1.42	0.27	18.34	1.23	0.28	0.43
6	20.33	-0.65	0.26	20.77	-0.83	0.26	0.47

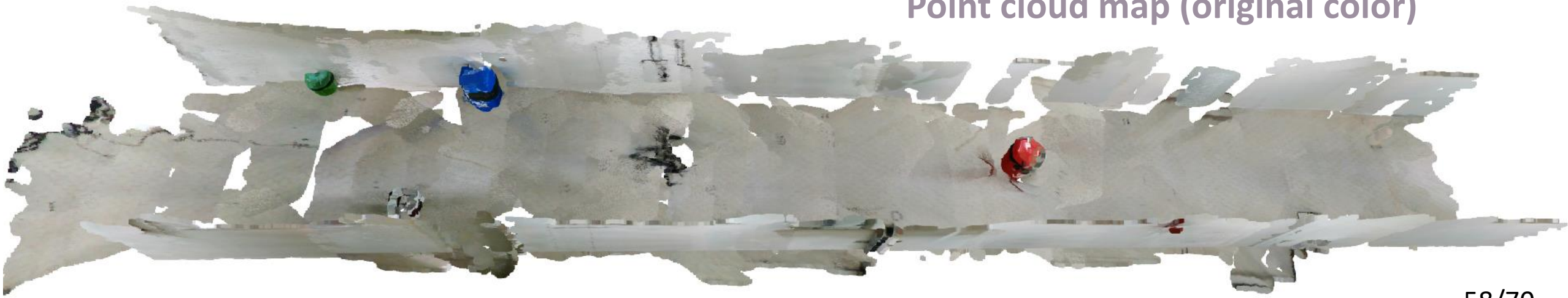
TABLE 5.4 – Plane feature error.

	n_a	n_b	n_c	$\hat{n}_a$	$\hat{n}_b$	$\hat{n}_c$	error
0	0	2.01	-0.10	-0.06	2.13	-0.10	0.13
1	0	0.00	-0.49	0.00	-0.01	-0.49	0.01
2	0	-1.03	-0.04	0.02	-0.90	-0.03	0.13

LMI corridor

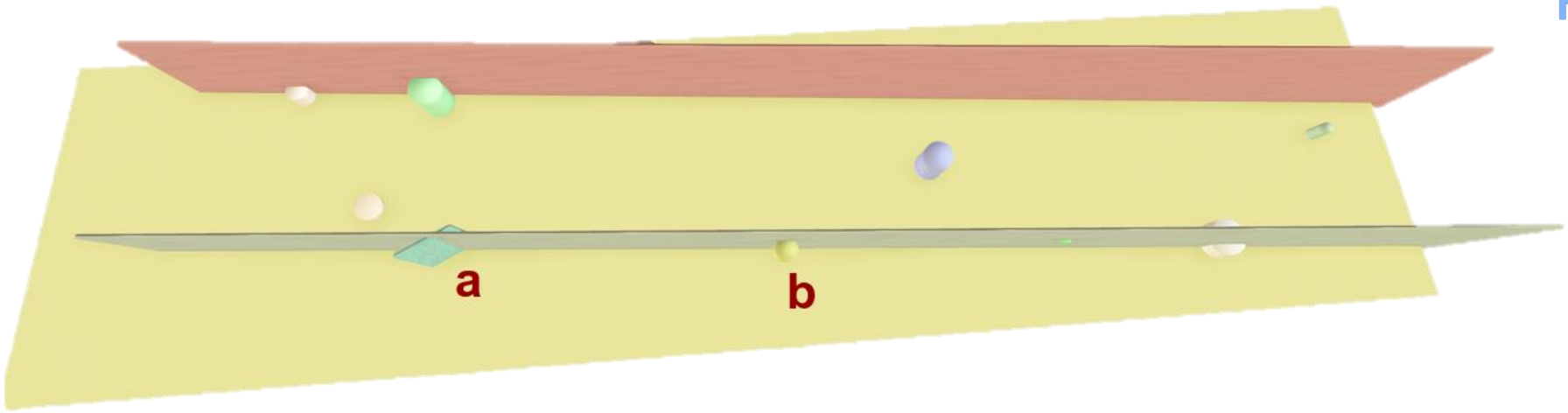
Point cloud map (colored by feature)

Point cloud map (original color)

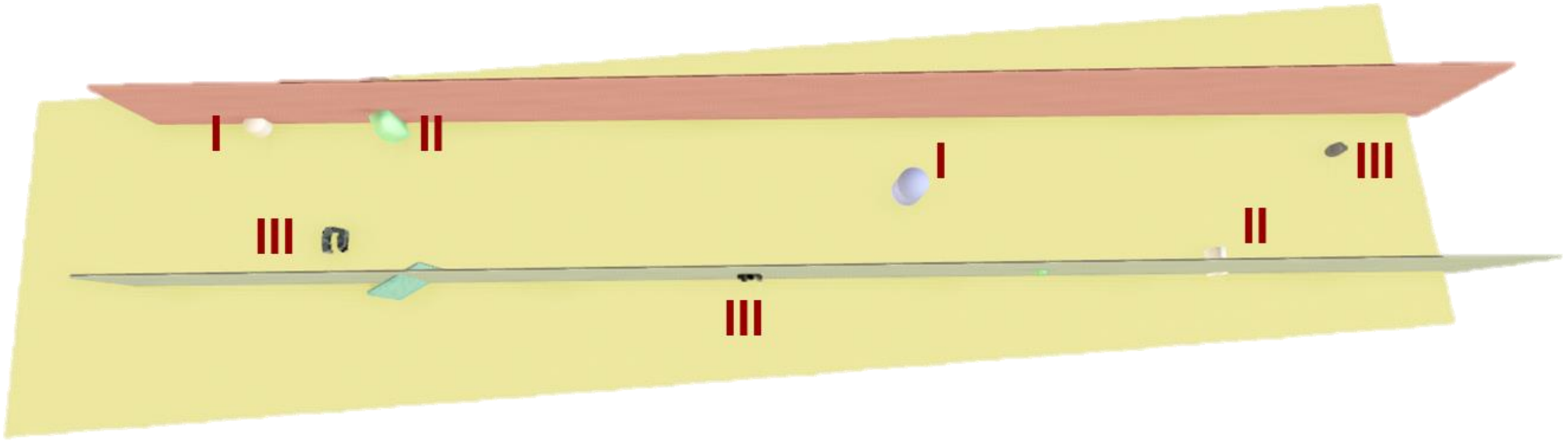


LMI corridor

High level map

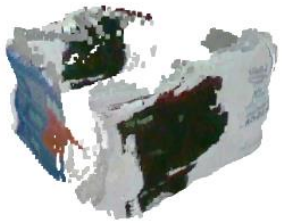


Hybrid map



(I) Cylinder
(II) Cuboid
(III) Point cloud

LMI corridor











Loop closure

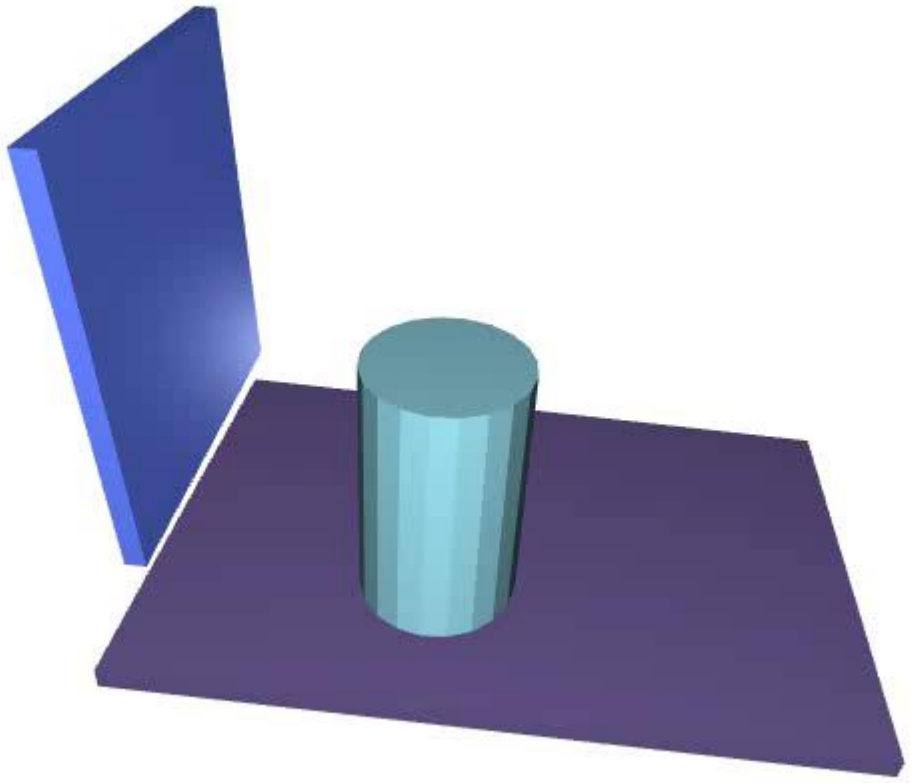
Loop closure

High level map

Point cloud map

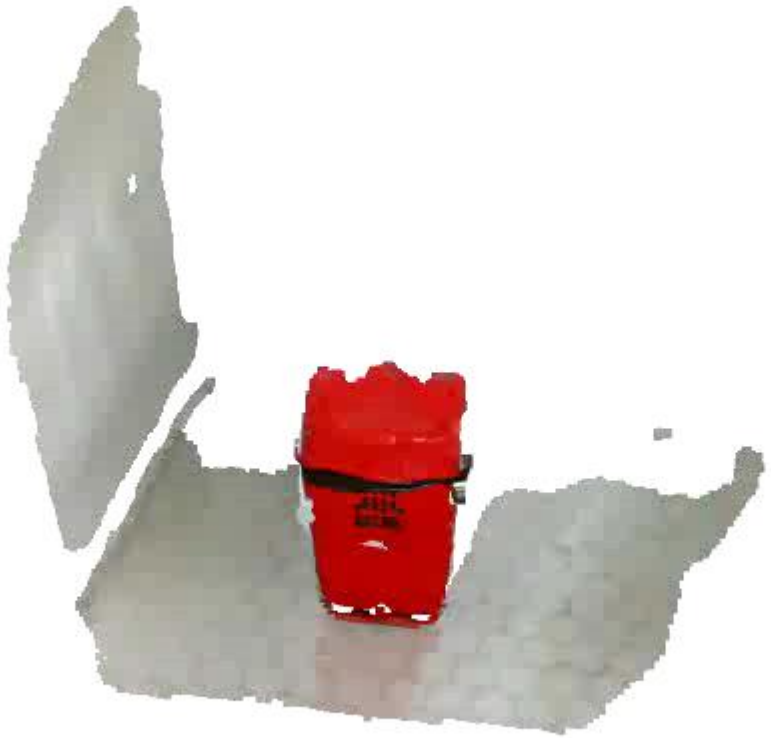
Loop closure

High level map



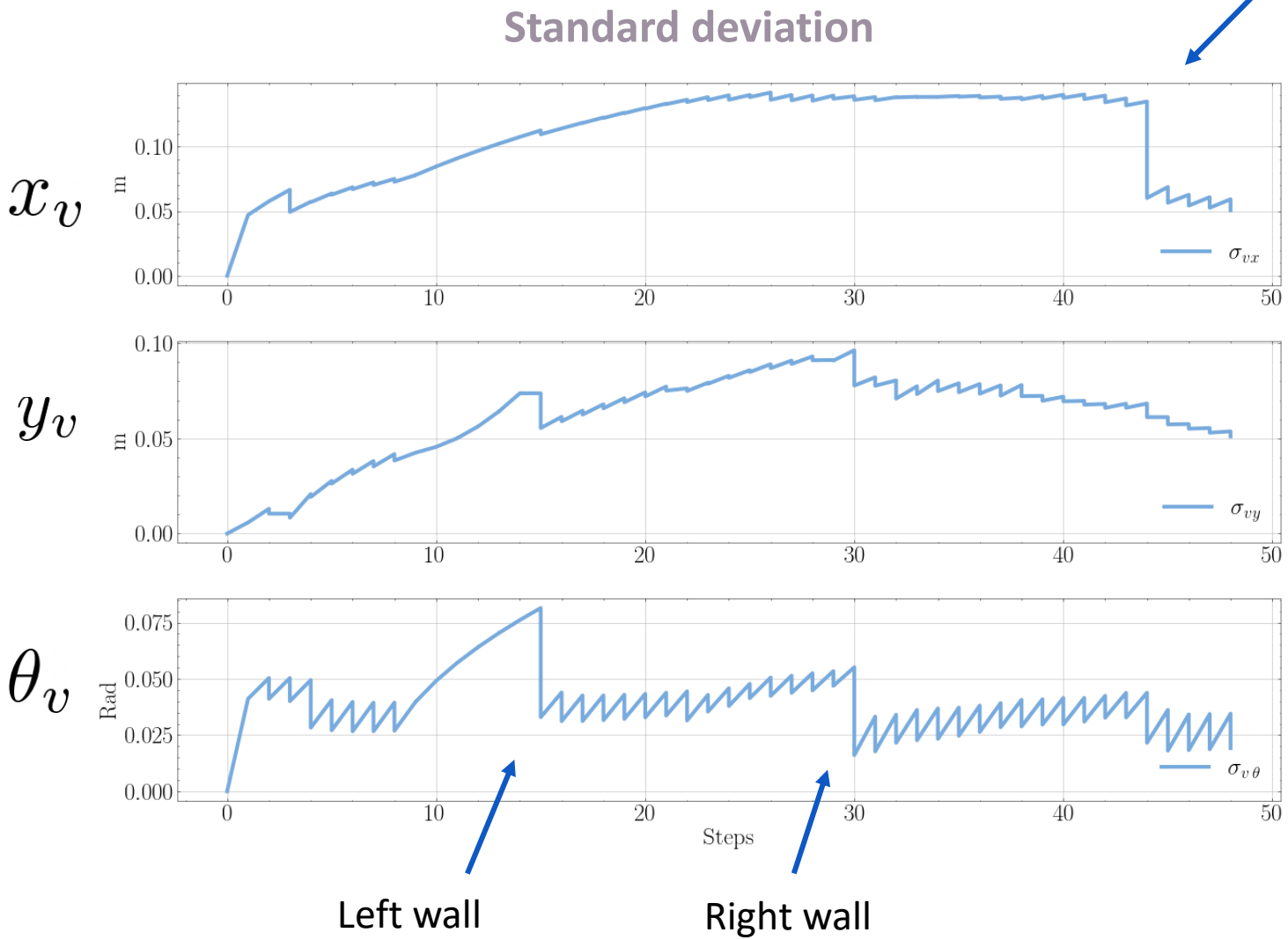
Loop closure

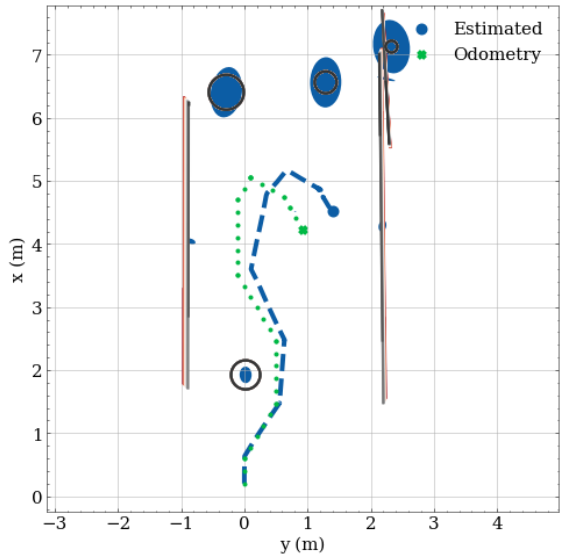
Point cloud map

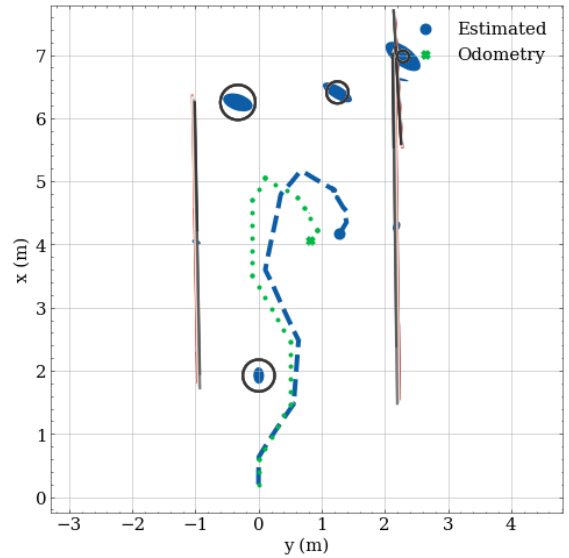


Loop closure

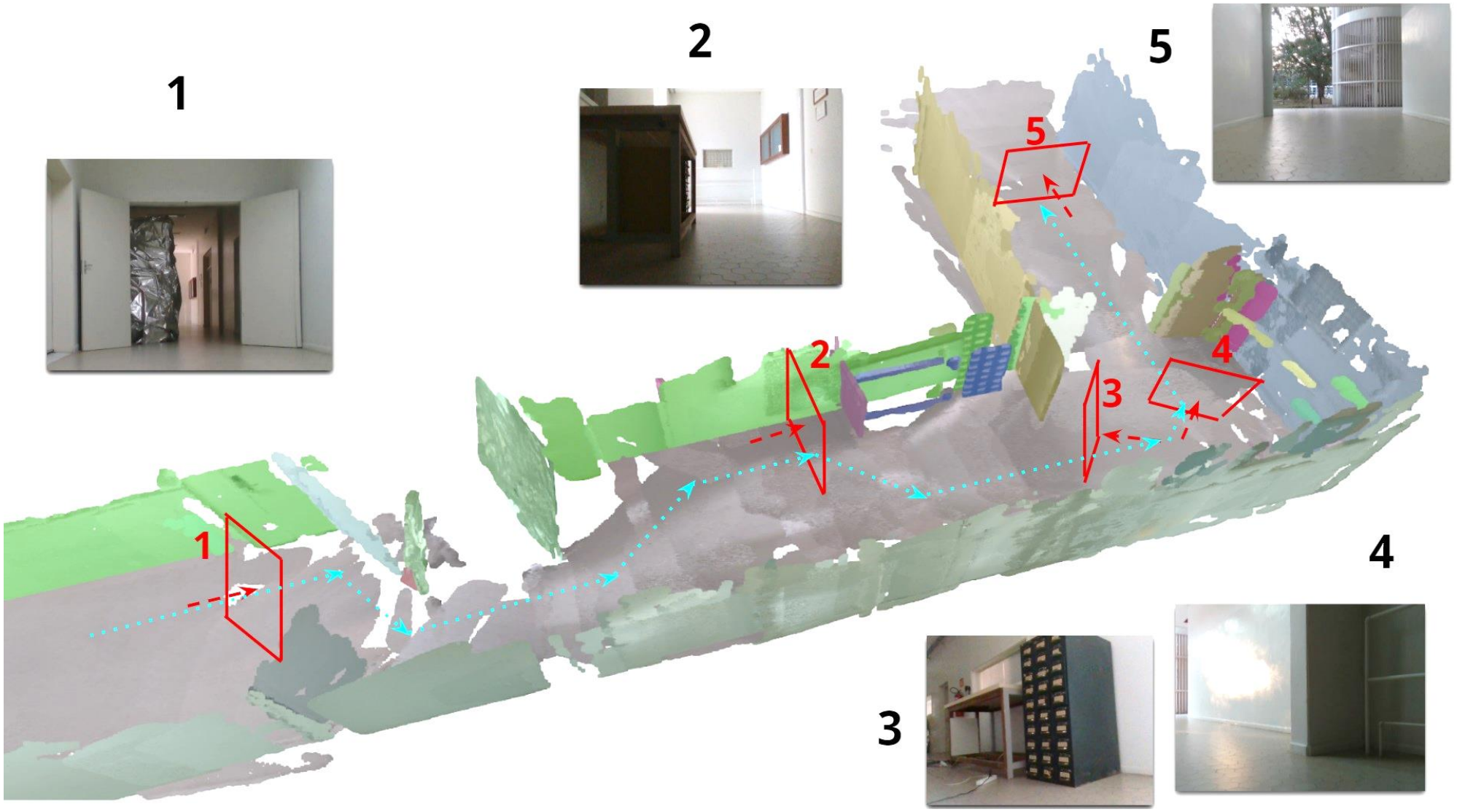
Red trash bin





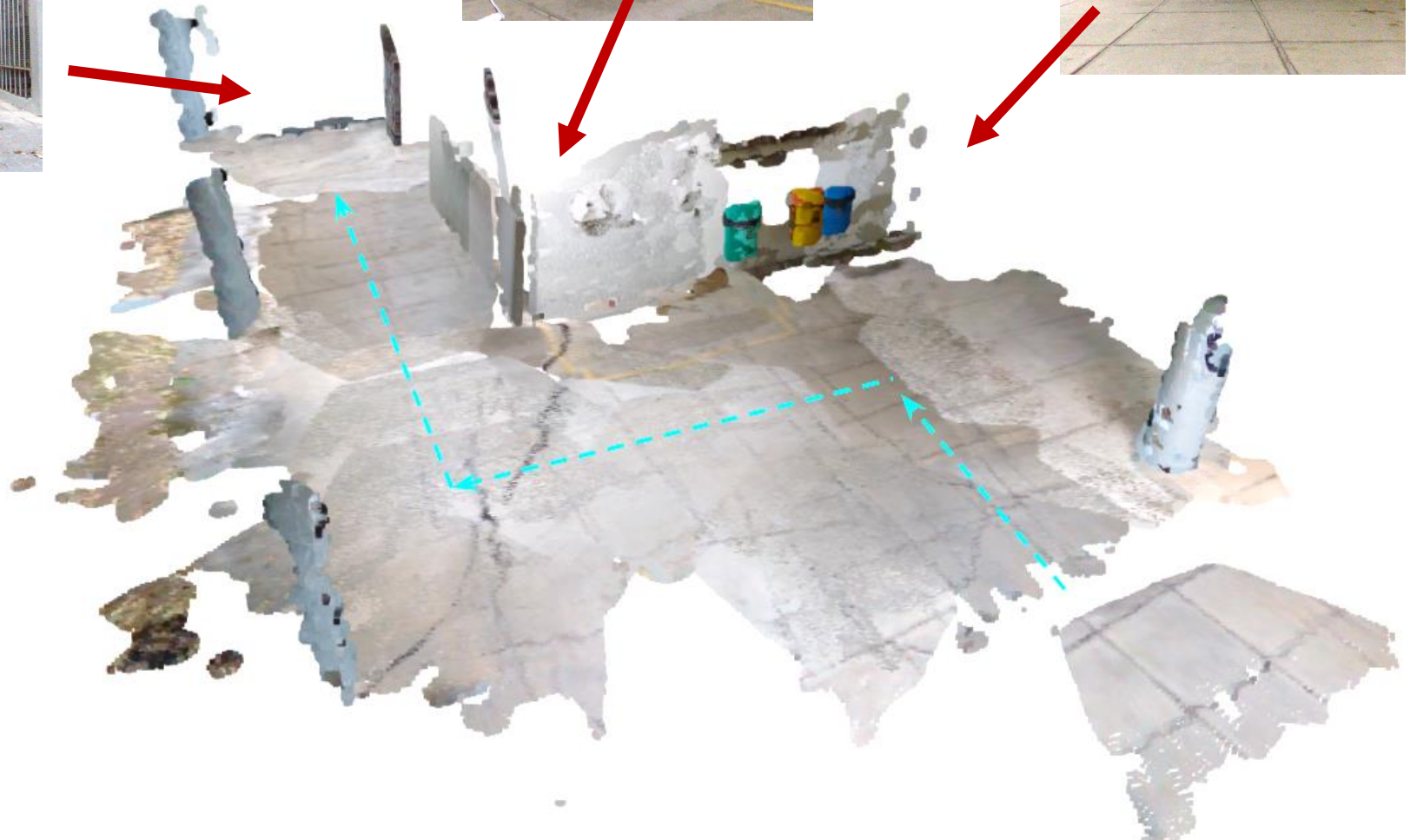


Corridor LMI part 2

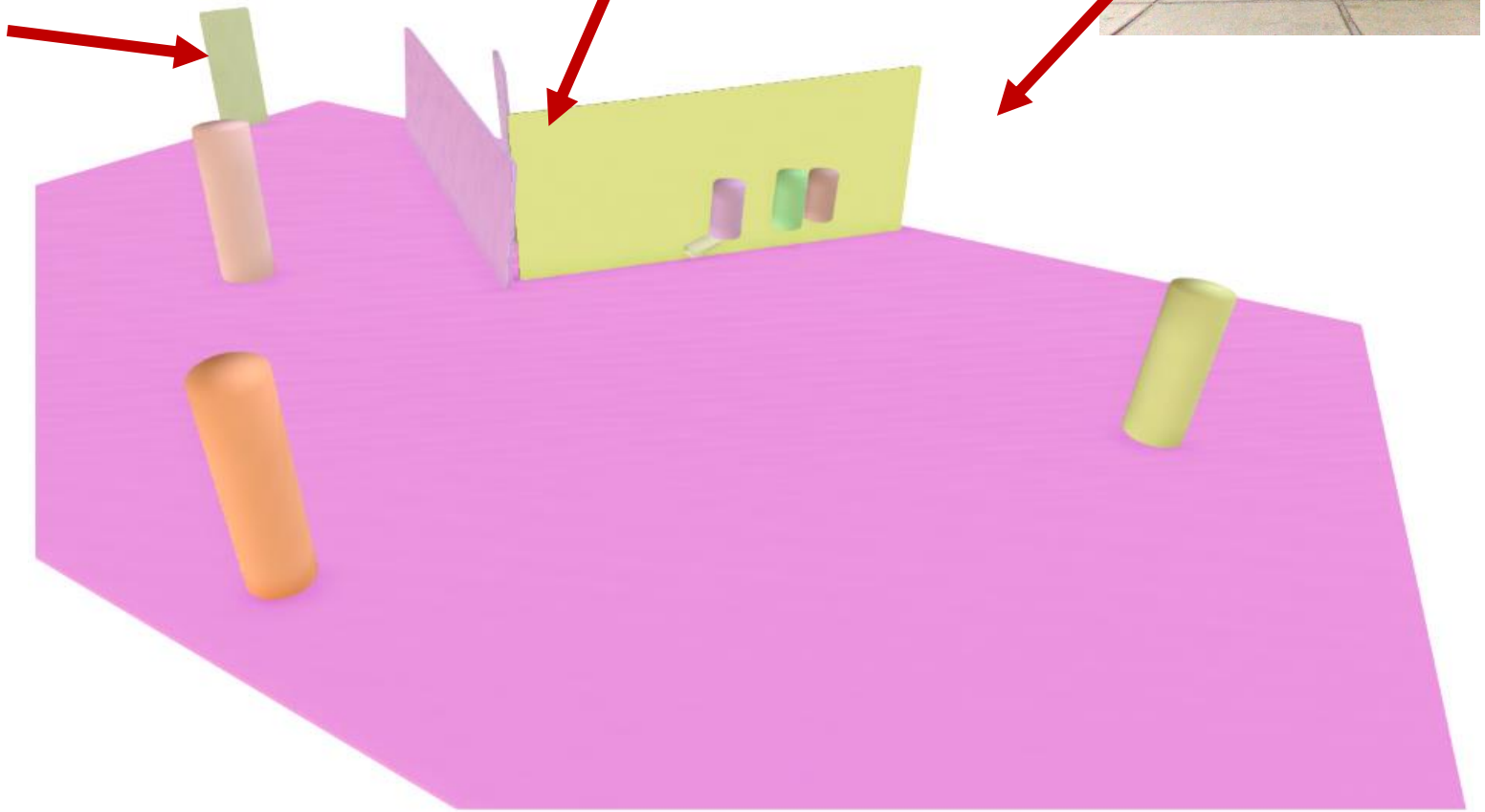


Recycle bin map

Point cloud map

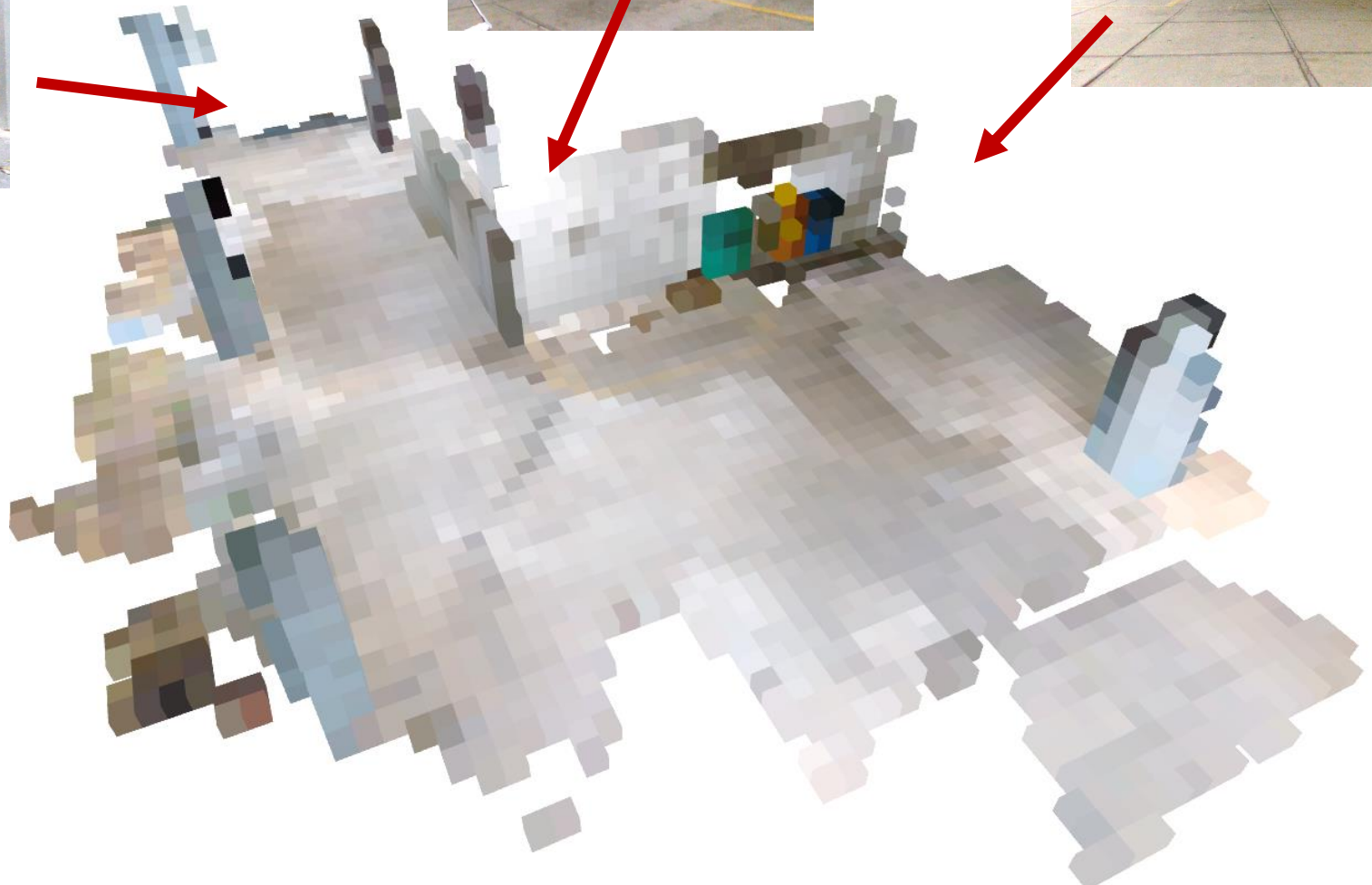


Recycle bin map



High level map

Recycle bin map



Voxel grid map

Conclusion

- ✓ Classical EKF-SLAM algorithm with a 3D realistic map,
- ✓ 85% Reduction of IAE error,
- ✓ Scene segmentation and geometry awareness,
- ✓ Hybrid map,
- ✓ Mapping representations comparisons.

Future works

- Experimental tests in larger environments,
- Data association in groups of features,
- Neural network for feature extraction.

Thank you!

Acknowledgments

Slide model by Pedro Henrique de Jesus (ITA)