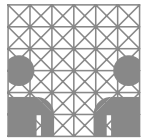


Robot Mapping: A Survey

Björn Sygo

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

09. December 2021

Outline

Mapping

- Robot has to acquire a spatial representation of its environment
- Needed for tasks like route planning, so the robot knows how to arrive at its target
- To accomplish this task, the robot often needs to localize itself in it
- Needs to be able to sense its environment
- Has to move around to gather the required data
- The produced map needs to be good enough for the required task, btu not perfect

Metric Map

- Most commonly associated with maps
- Describe the geometrical properties of an environment
- Each object saved with precise coordinates
- Can be continuous or grid-based

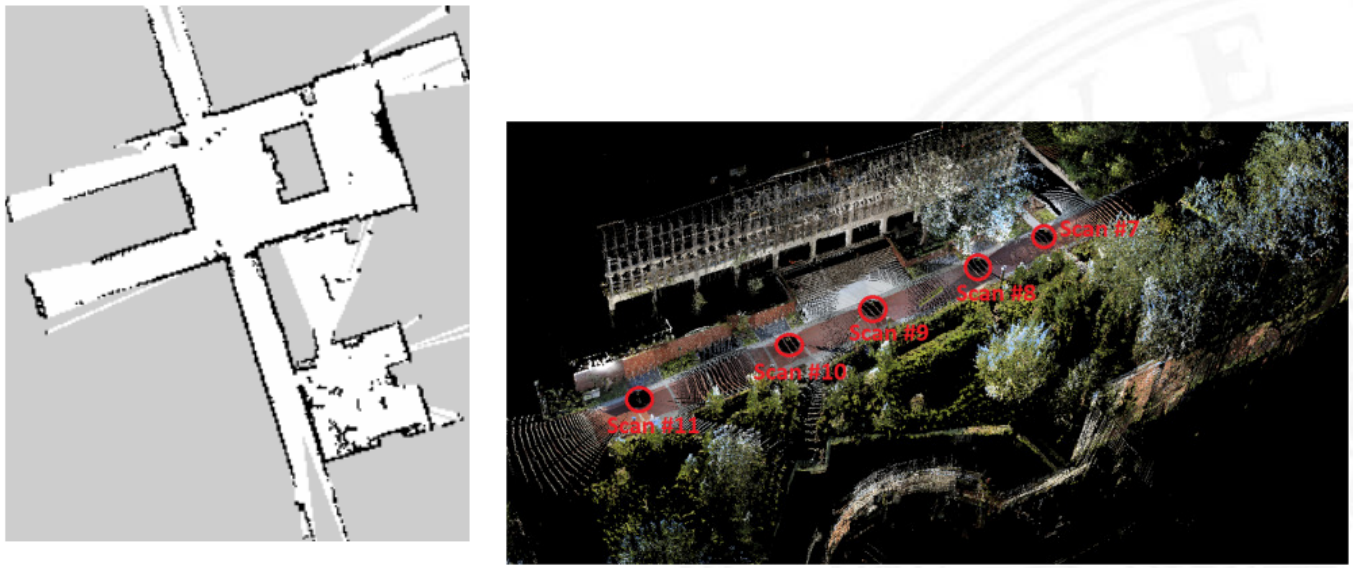


Figure 1: Different types of metric maps. Left: Grid based metric map. [5] Right: 3D continuous metric map. [4]

Metric Map Pros & Cons

- Easy to plan a path
- Intuitive visualization as pictures or 3D plots
- Dense representation saves a lot of unnecessary information
- Computationally expensive to create

Topological Map

- Map consists only of distinguishable points of interest
- Usually graphs connect points to give possible routes
- For humans these points would be landmarks like mountains or specific buildings
- Robots can't easily perceive these landmarks so most commonly (Image-)features are used instead

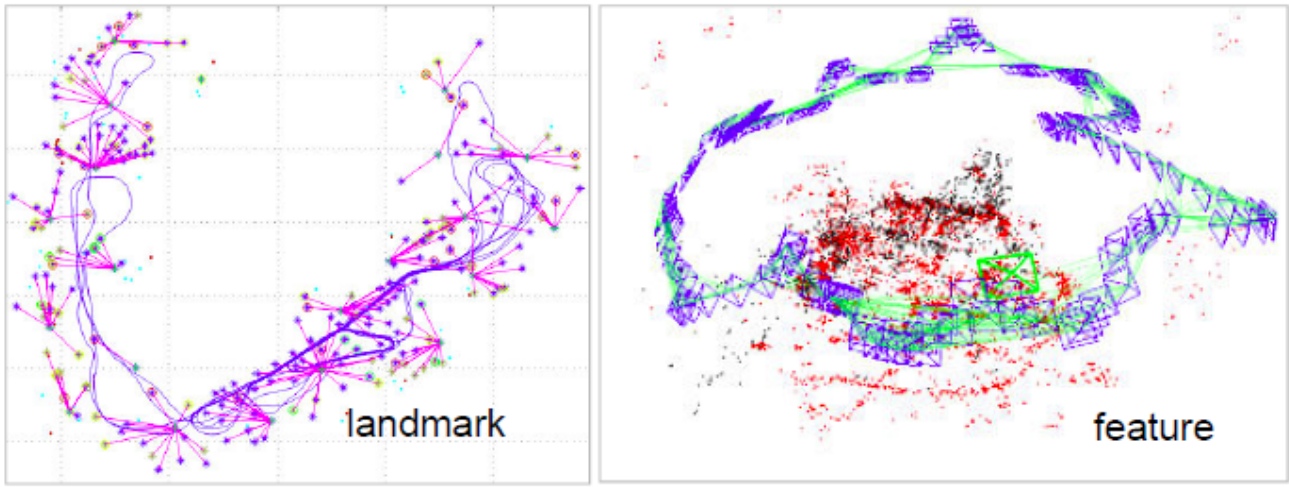


Figure 2: Two types of topological maps. Left: Landmark representation. Right: Feature representation. [13]

Topological Map Pros & Cons

- Sparse representation allows efficient map saving
- Few saved points allow fast localization
- Environment needs to have enough landmarks/features
- They need to be easily differentiable

- Stands for Simultaneous Localization And Mapping
- Robot has to move during map creation
- Also needs to know where it is to create the map
- Need to both at the same time
- Most common application for mapping

Probabilistic Mapping

- All measurements have noise
- Algorithms have to be able to handle it
- Mapping problem is highly complex
- Therefore, almost all algorithms are probabilistic
- Some explicit with mathematical derivations other implicit

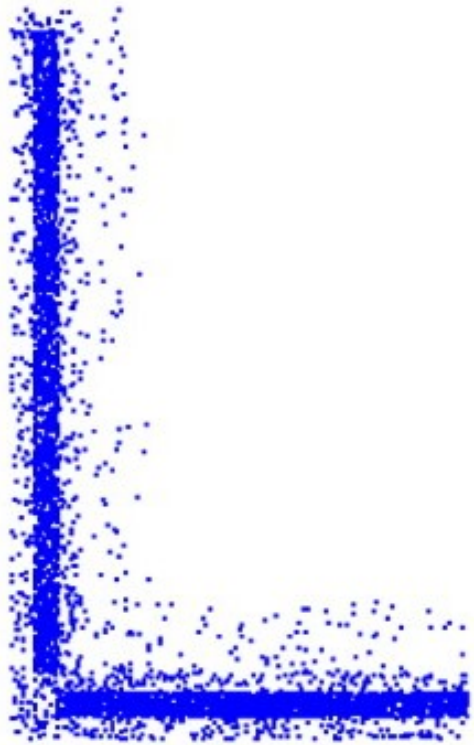


Figure 3: Noisy point-cloud data of a corner. [9]

Bayes Filter Introduction

- Basis for almost all probabilistic mapping algorithms
- Extension of the bayes rule to incorporate temporal changes
- Used to get the most probable state and map given previous measurements

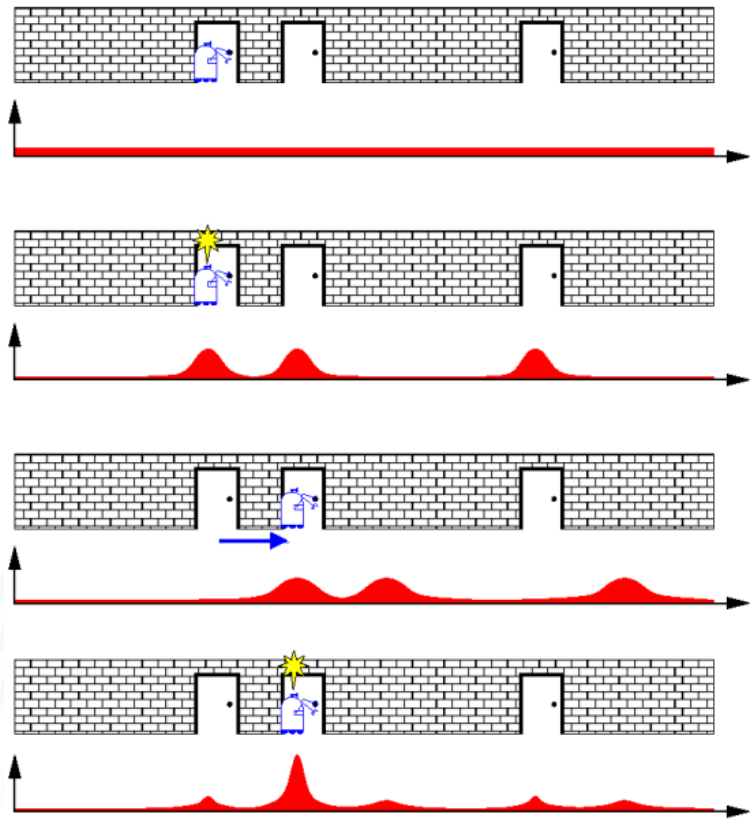


Figure 4: Robot driving between doors example for bayes filter. [8]

Bayes Filter Formula

Bayes Rule

$$p(x|d) = \eta p(d|x)p(x)$$

$$p(s_t, m|z^t, u^t) = \eta p(z_t|s_t, m) \int p(s_t|u_t, s_{t-1}) p(s_{t-1}, m|z^{t-1}, u^{t-1}) ds_{t-1}$$

s_t : robot pose at time t

m : static map

z^t : all sensor measurements until time t

z_t : sensor measurement at time t

u^t : all robot motion commands until time t

u_t : robot motion command in interval [t-1,t)

Bayes Filter Posterior

Posterior

$$p(s_t, m|z^t, u^t)$$

s_t : robot pose at time t

m : static map

z^t : all sensor measurements until time t

u^t : all robot motion commands until time t

- Goal probability to estimate
- Probability of a specific map and state at time t given all measurements and motion commands until t
- Maximise to find the most probable state and map for one point in time

Bayes Filter Perceptual Model

Perceptual Model

$$p(z_t|s_t, m)$$

s_t : robot pose at time t

m : static map

z_t : sensor measurement at time t

- Specifies the probability of a sensor measurement given a specific robot state and map
- Usually time invariant, so often written without time dependency
- This model needs to be specified to describe how sensor measurements are generated

Bayes Filter Motion Model

Motion Model

$$p(s_t | u_t, s_{t-1})$$

s_t : robot pose at time t

m : static map

u_t : robot motion command in interval [t-1,t)

- Specifies the probability of a specific state given the previous robot state and the motion control issued between
- Usually time invariant, so often written without time dependency
- This model needs to be specified to describe how probable the robot executes its motion commands correctly

Bayes Filter

$$p(s_t, m|z^t, u^t) = \eta p(z_t|s_t, m) \int p(s_t|u_t, s_{t-1}) p(s_{t-1}, m|z^{t-1}, u^{t-1}) ds_{t-1}$$

s_t : robot pose at time t

m : static map

z^t : all sensor measurements until time t

z_t : sensor measurement at time t

u^t : all robot motion commands until time t

u_t : robot motion command in interval [t-1,t)

- Each state has to incorporate all unknown quantitites that may influence measurements
- Assumes independent noise which is usually not the case
- Can't be solved by digital computers so further assumptions need to be made

Occupancy Grid Map

- Probabilistic metric map representation
- Most commonly mentioned for 2D, 3D version refered to as Voxelgrid
- Each grid cell has a binary occupancy value
- Can be seen as sample from probability distribution
- Relatively robust and easy to implement
- Basic algorithm assumes known poses
- Binary bayes filter can then be used to calculate occupancy probability for each grid cell

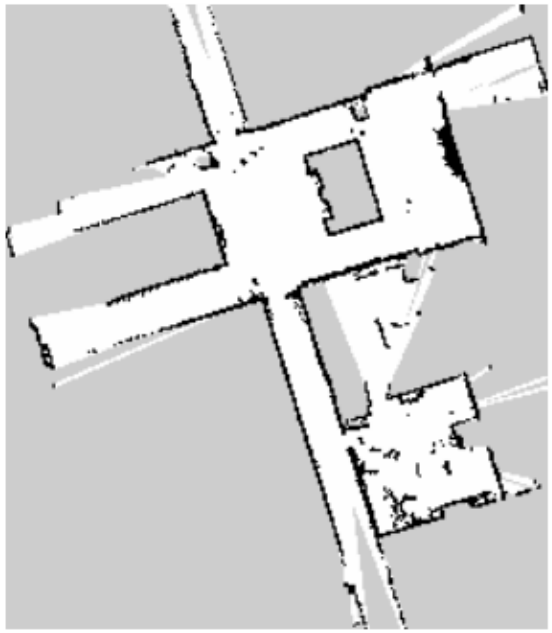
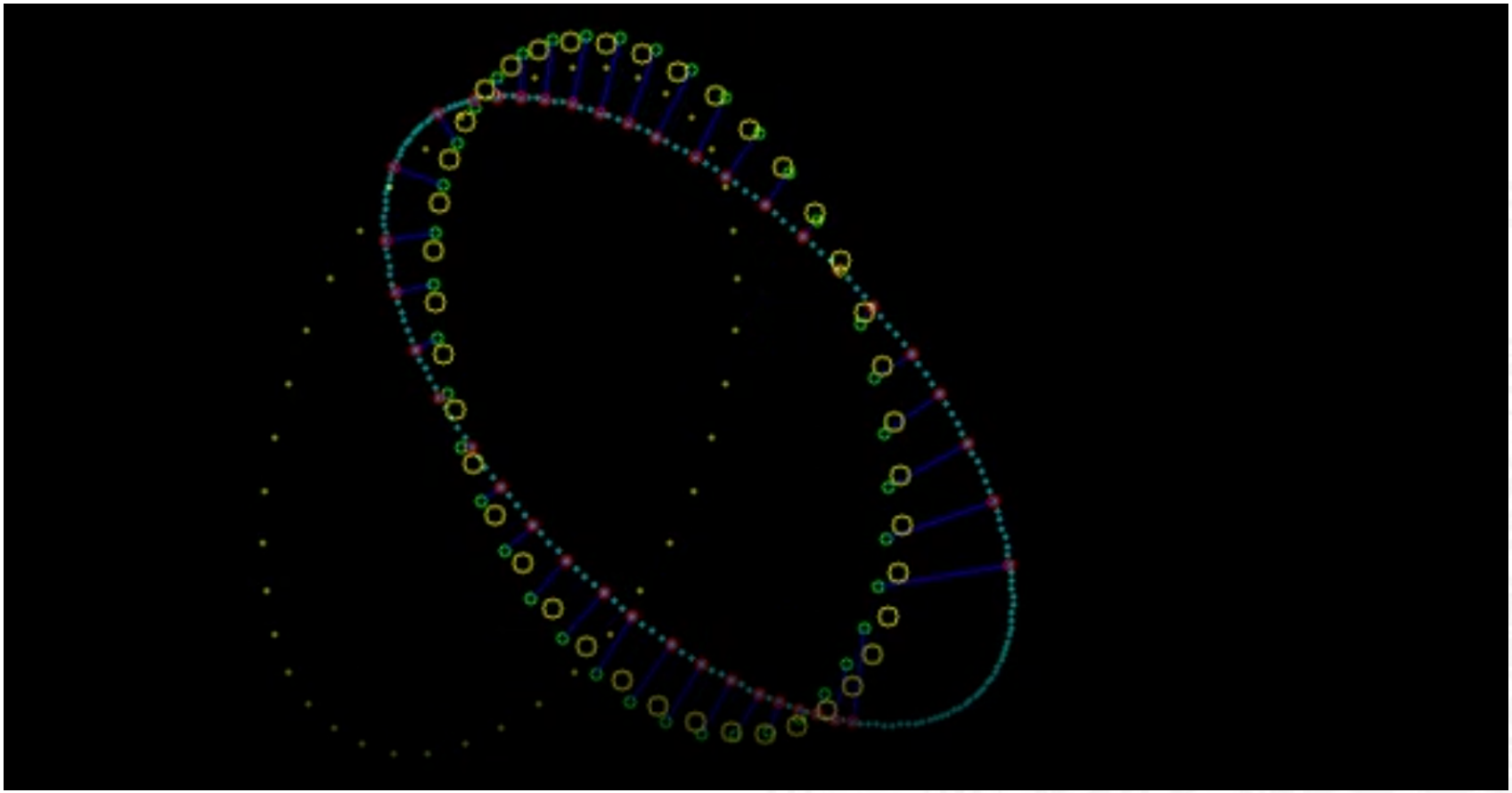


Figure 5: Occupancy Grid Map. [5]



Scan Matching

- Scan is a sequence of scan points
- Try to match a scan to a previous one
- Need a previous position, previous scan and new scan
- Output rigid transformation so scans are aligned

$$x_t^* = \underset{x_t}{argmax}\{p(z_t|x_t, m_{t-1})p(x_t|u_{t-1}, x_{t-1}^*)\}$$

- Often alternative to vison based methods of localization and mapping

Hector-SLAM video

Overview

- Uses LIDAR sensor for its 2D SLAM
- Represents the Map as an Occupancy Grid Map
- Uses gradient ascent scan matching for mapping

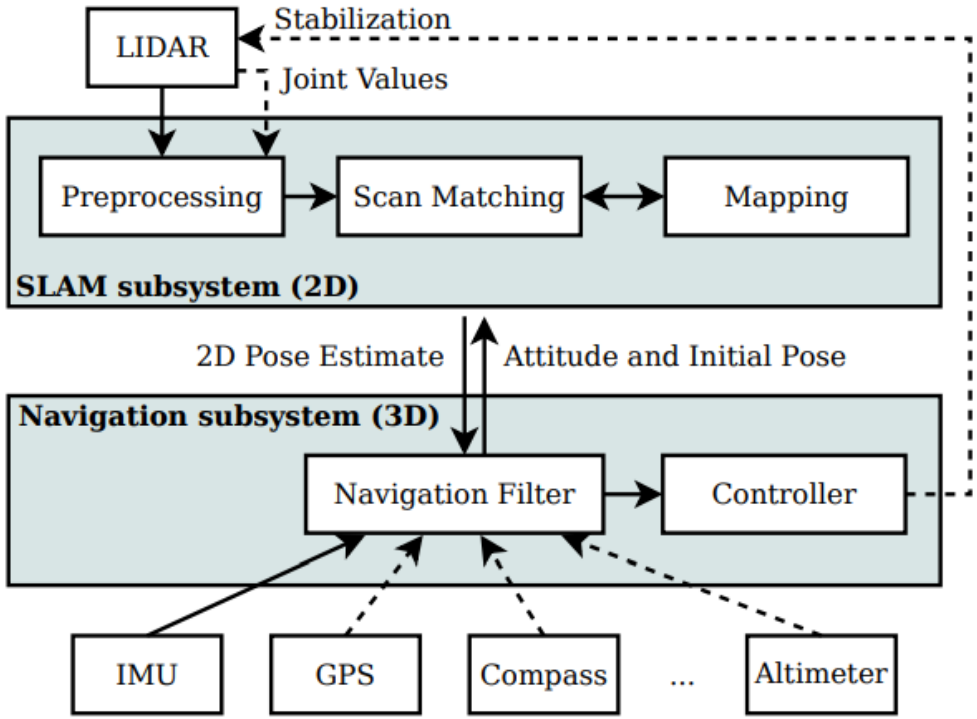


Figure 6: Overview of the hector-SLAM architecture. [5]

Implementation Map

- Coordinate system initialized at the starting point
- Multiple Maps with different resolutions to mitigate local minima
- Each generated individually instead of downsampling
- Scan alignment starts at the coarsest level

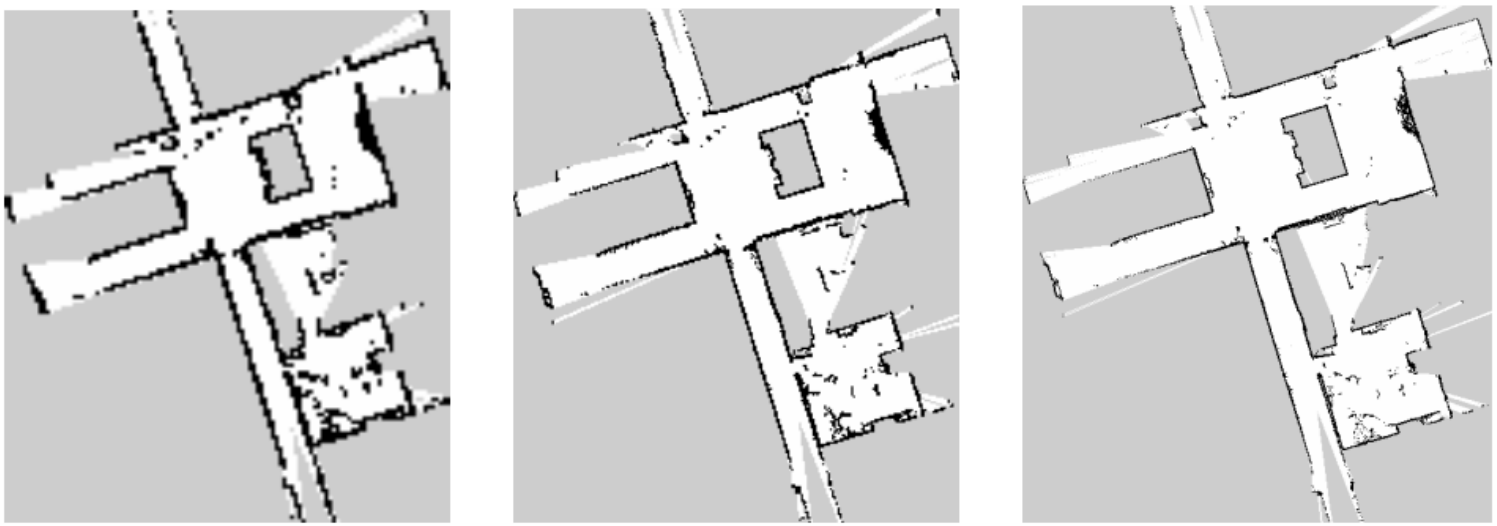


Figure 7: Example of a hector-SLAM occupancy grid map with increasing resolution from left ro right. [5]

Implementation Scan Matching

- Rely on accurate and fast scanning LIDAR sensors
- Try to find best partial alignment with existing map
- Unnecessary to do additional exhaustive pose search
- Use Gauss-Newton approach to minimize the following equation

Implementation Scan Matching

$$\xi^* = \underset{\xi}{argmin} \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2$$

$$S_i(\xi) = \begin{pmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

$M(x)$: occupancy value of point x

$\begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix}$: coordinates of scan point i

$\xi = (p_x, p_y, \psi)^T$: rigid transformation of the robot pose

Loop Closure

- Loop Closure: Detection and fusion of loops in the mapped environment
- Hector-SLAM has no explicit loop closure mechanism
- Accuracy high enough to close small loops
- Larger loops might not be closed due to accumulated error

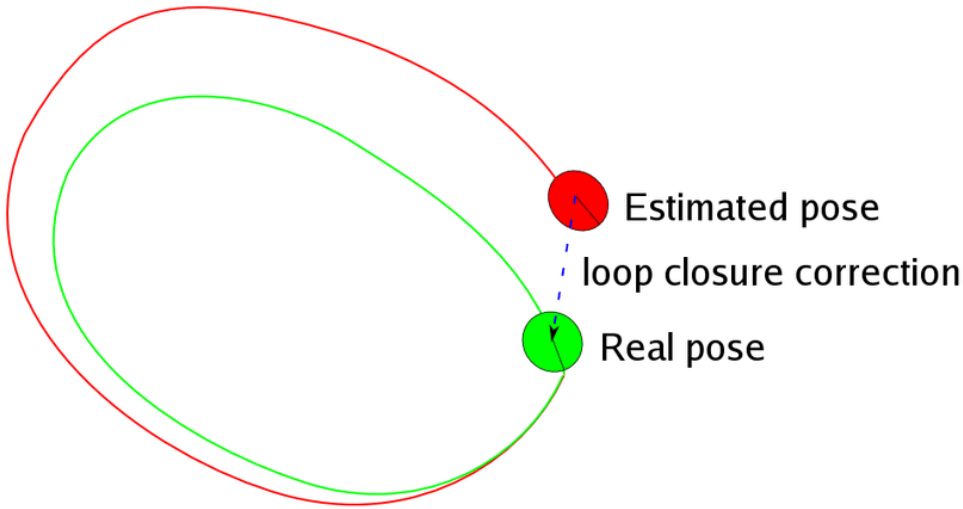
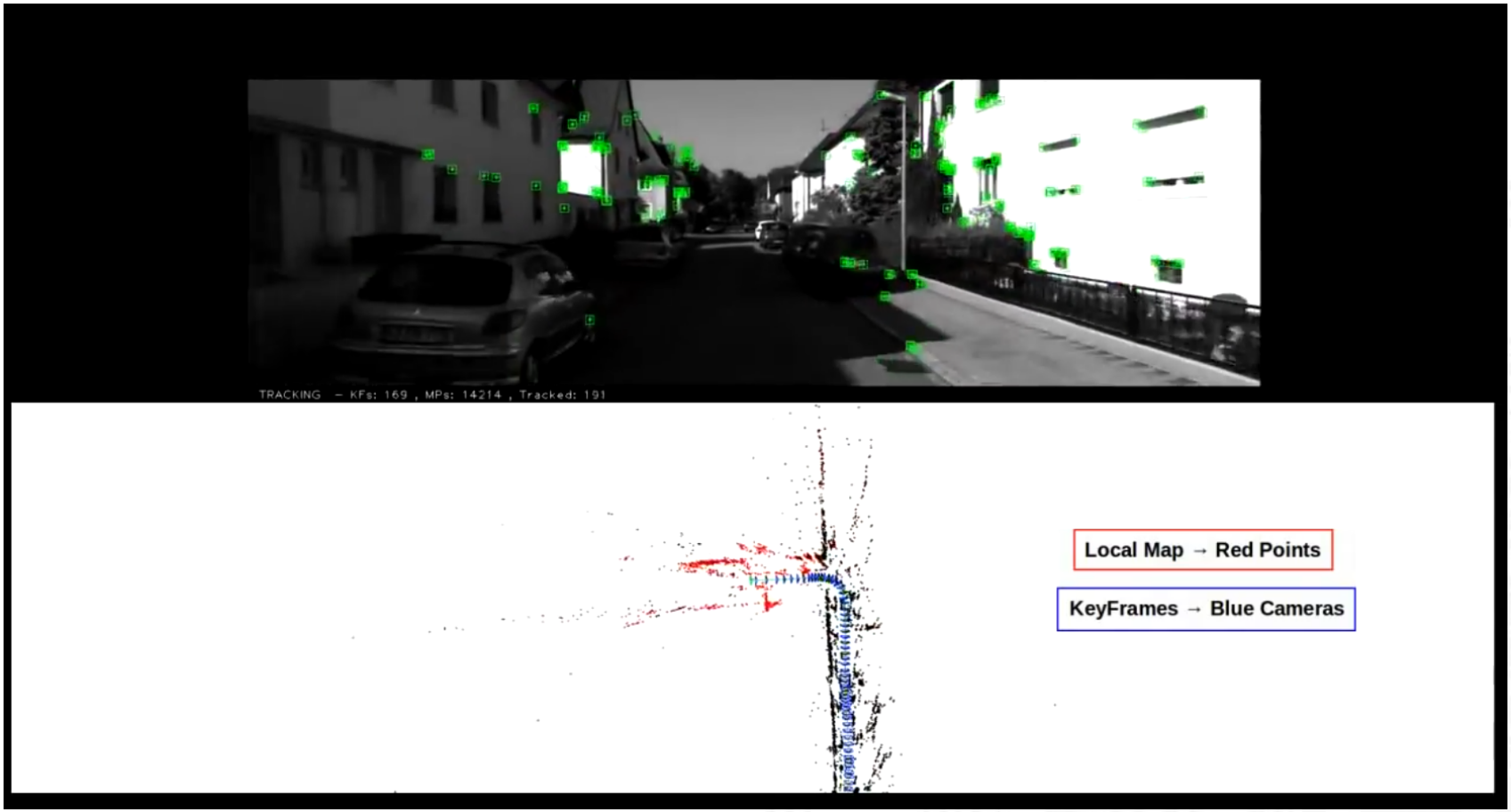


Figure 8: Basic principle of loop closure. [1]

ORB-SLAM video



Overview

- Image feature based mapping algorithm
- Keyframe based approach based on PTAM
- Uses Bundle Adjustment for localization

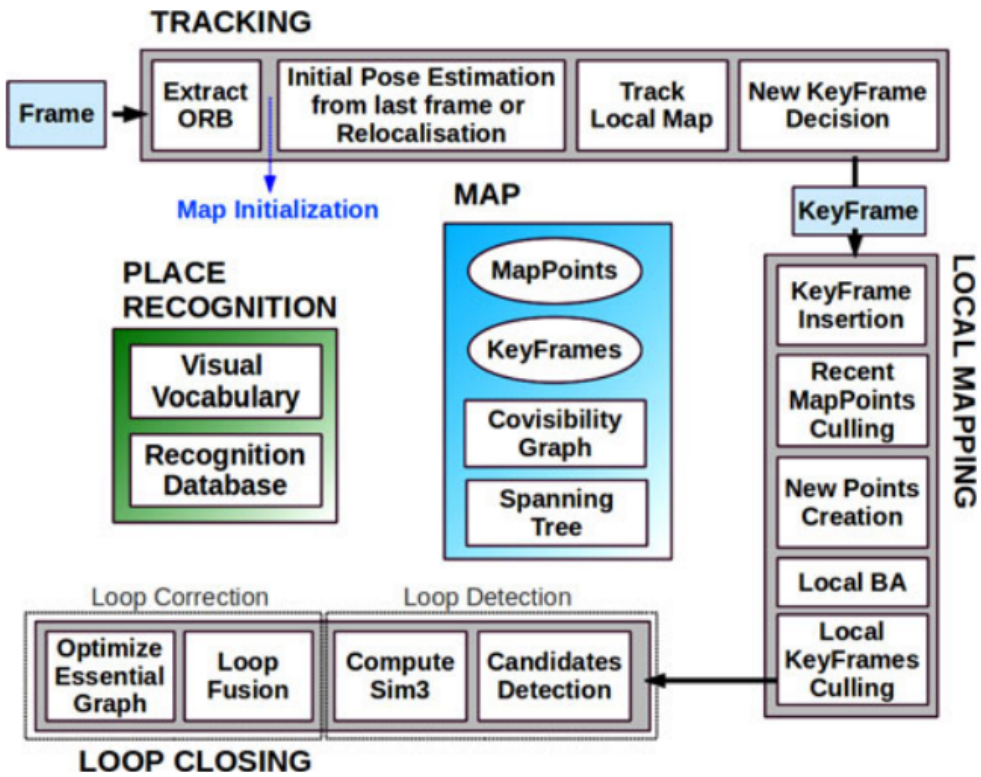


Figure 9: Overview of the ORB-SLAM architecture. [6]

Excursion: ORB-Features

- Image features called Oriented Fast and Rotated Brief features
- Name derived from enhanced commonly used feature descriptors
- Feature descriptors extract points from an image that are unique in their region like corners
- ORB-features have more invariance to viewpoint than other common features
- They are represented as 256-bit descriptors

Figure 10: Example image with detected SIFT-features marked.

Map Representation

- Map points consist of following:
  1. 3D position in the world coordinate system
  2. Viewing direction: mean unit vector of all viewing directions
  3. ORB descriptor with minimal hamming distance to all other associated descriptors
  4. Maximum and minimum visible distance of point according to ORB limitations
- Keyframes contain the following:
  1. Camera pose: rigid body tranformation from world frame to camera pose
  2. Camera intrinsics
  3. All ORB features from that frame with their matched map point if existing
- For each keyframe, a bag of words representation is also computed and saved
- Keyframes then saved in the following graphs

Mapping Graphs

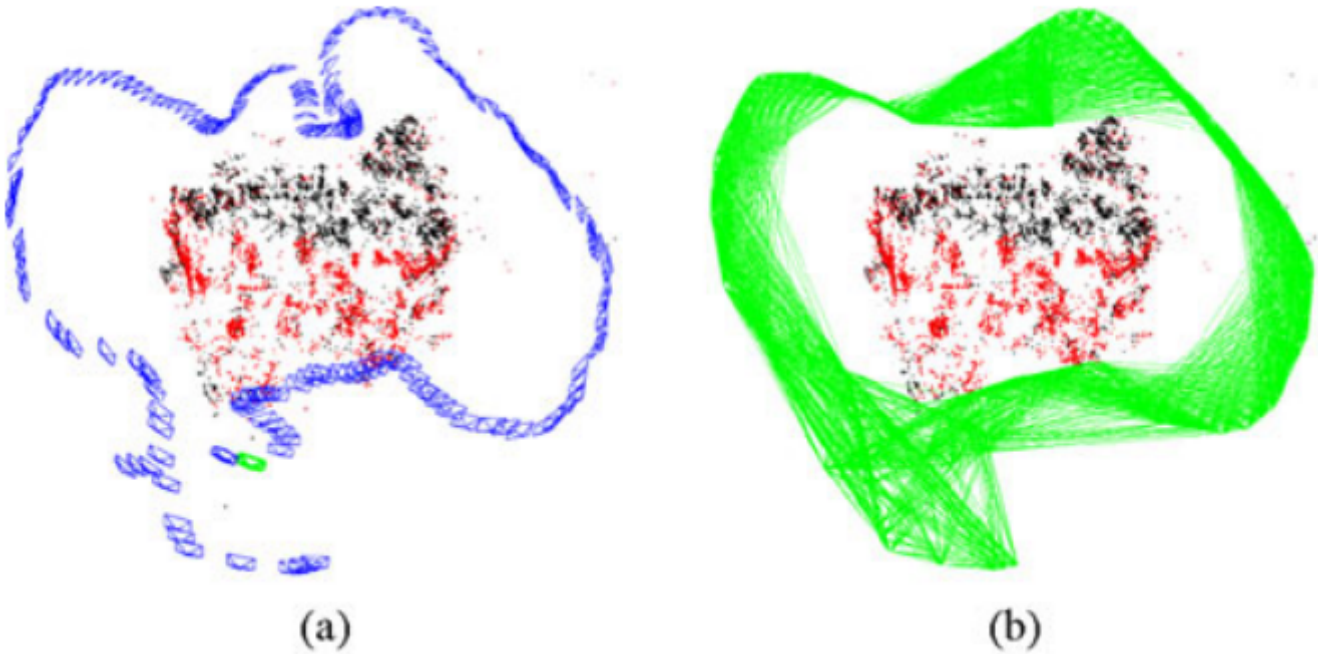
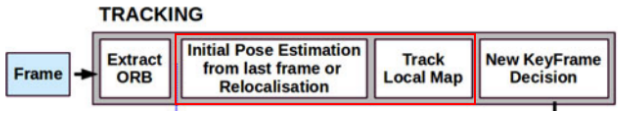


Figure 11: Different graphs used in the ORB-SLAM implementation for mapping. (a): Keyframes. (b): Covisibility graph. Red & black: map points with red the currently visible map points. [6]

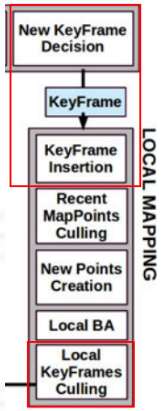
Tracking



- Extract ORB-descriptors on different scale levels
- Predict current position based on motion commands and search for map points from last frame
- If not enough were found, compute bag of words representation and search again from closest matching keyframe candidate
- Optimize the position estimate by searching the local map for map point correspondence
- Local map: current frame, keyframes that share map points with current frame and their neighbors in the covisibility graph
- Optimization done through motion-only bundle adjustment
- Discard not visible map points and find best fitting ORB-feature in the current frame for each remaining map point

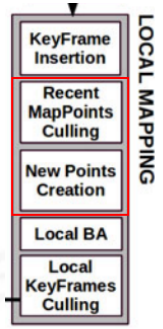
Keyframe Handling

- Focus to insert keyframes fast and cull later
- Frame has to fulfill different criteria:
  1. At least 20 frames must have past from global relocalization
  2. Local mapping is idle or 20 frames from last keyframe insertion
  3. Frame has at least 50 map points
  4. less than 90% of points from last keyframe
- Add new node to covisibility graph
- Insert edges to keyframes that have the same map points
- Compute bag of words representation and save
- Redundant keyframes later culled if 90% of its map points can be seen on same or finer scale in at least three other keyframes



Map Point Handling

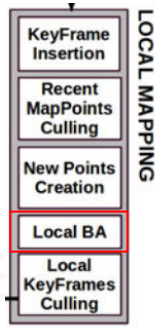
- Map points created when ORB-features triangulated from keyframe neighbors in covisibility graph



- Initially only from two keyframes but matches searched for in connected keyframes
- Map Points tested during first three keyframes it should be visible in
- Deleted if not found often enough by tracking or not visible from enough keyframes

Local Bundle Adjustment

- Bundle Adjustment: Refine coordinates and camera parameters based on different viewpoints
- Goal to optimally bundle light rays emerging from each observed feature
- Usually implemented minimising a predefined error term
- Can be seen as maximum likelihood estimator
- Local: Optimize current keyframe and neighbors in covisibility graph
- Other keyframes that see points also in these keyframes are included but remain fixed



Local Bundle Adjustment Formula

Error Term

$$e_{i,j} = \mathbf{x}_{i,j} - \pi_i(T_{iw}, X_{w,j})$$

$\mathbf{x}_{i,j}$: reprojection of matched keypoints

T_{iw} : keyframe poses regarding world reference w

$X_{w,j}$: 3D coordinates of map point j regarding world reference w

Projection Function

$$\pi_i(T_{iw}, X_{w,j}) = \begin{pmatrix} f_{i,u}\frac{x_{i,j}}{z_{i,j}} + c_{i,u} \\ f_{i,v}\frac{y_{i,j}}{z_{i,j}} + c_{i,v} \end{pmatrix}$$

$[x_{i,j}, y_{i,j}, z_{i,j}]$: 3D coordinates of map point j in keyframe i

$(f_{i,u}, f_{i,v})$: focal length of camera i

$(c_{i,u}, c_{i,v})$: principal point of camera i

Local Bundle Adjustment Formula

Map Point Projection

$$[x_{i,j}, y_{i,j}, z_{i,j}]^T = R_{iw} X_{w,j} + t_{iw}$$

R_{iw} : rotation part of T_{iw}

t_{iw} : translation part of T_{iw}

Cost Function

$$C = \sum_{i,j} \rho_h(e_{i,j}^T \Omega_{i,j}^{-1} e_{i,j})$$

ρ_h : Huber loss function

C : cost to be minimized

$\Omega_{i,j}^{-1}$: covariance matrix of scale corresponding to the keypoint

Loop Closure Graphs

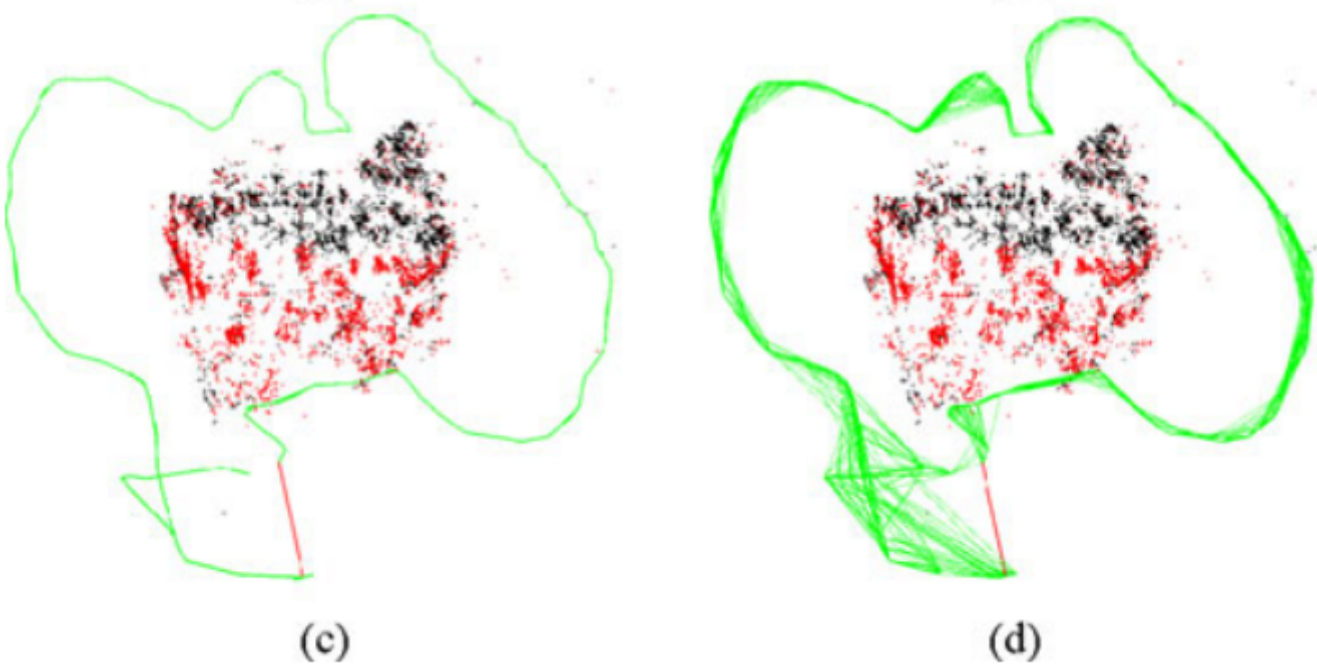
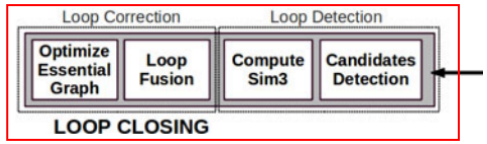


Figure 12: Different graphs used in the ORB-SLAM implementation for loop closure. (c): Spanning Tree. (d): Essential Graph. Red & black: map points with red the currently visible map points. [6]

Loop Closure

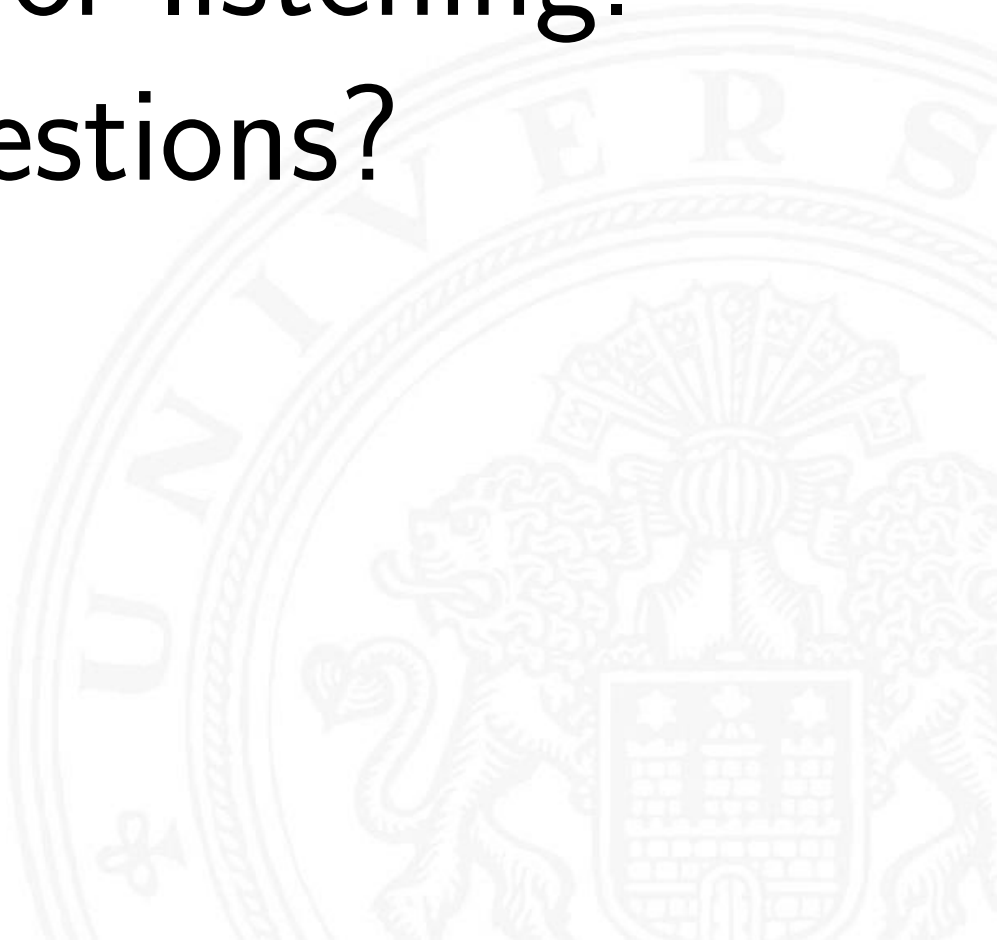


- Detect loop candidates by computing similarity of bag of words representation of current frame with covisibility neighbors
- Search for other keyframes that are as similar and use them as loop candidates if found
- Loop candidates only used if at least three are found consecutively
- Then compute similarity transform between loop candidate keyframes and accept loop if similarity transform is supported enough
- Fuse duplicate map points and correct keyframe pose with similarity transform
- Propagate the correction to the covisibility neighbors and optimize the entire essential graph

Summary

- Mapping is a very common and important part of many robotic applications
- There are many different implementations and representations
- Many parts not mentioned like 3D representation and kalman-filters
- Still an active field of research with increasing robustness and solving the problem for dynamic environments
- Several other fields are connected to mapping like exploration or scan matching, which weren't really explained either
- Still, I hope this was a good starting point for anyone interested to look further into the topic

Thank you for listening.
Any questions?



Sources

[1] Ahmed Abdelbaki, Maren Bennewitz, and Reza Sabverzavi. "ConvNet Features for Lifelong Place Recognition and Pose Estimation in Visual SLAM". PhD thesis. Mar. 2018. DOI: 10.13140/RG.2.2.10816.89607.

[2] *Handheld Mapping System in the RoboCup 2011 Rescue Arena*. URL: https://www.youtube.com/watch?v=F8pdObV_df4&list=PL0E462904E5D35E29 (visited on 12/07/2021).

[3] *ICP 2d Ellipse*. URL: https://www.youtube.com/watch?v=tfckXoa-wRQ (visited on 12/07/2021).

Sources (cont.)

[4] Pileun Kim, Jingdao Chen, and Yong K. Cho. "SLAM-driven robotic mapping and registration of 3D point clouds". In: *Automation in Construction* 89 (2018), pp. 38–48. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2018.01.009. URL: https://www.sciencedirect.com/science/article/pii/S0926580517303990.

[5] Stefan Kohlbrecher et al. "A flexible and scalable SLAM system with full 3D motion estimation". In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. 2011, pp. 155–160. DOI: 10.1109/SSRR.2011.6106777.

Sources (cont.)

[6] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[7] *ORB-SLAM in the KITTI dataset (Sequence 00)*. URL: https://www.youtube.com/watch?v=8DISRmsO2YQ (visited on 12/07/2021).

[8] *Probabilistic Robotics: Discrete Filters and Particle Filters*. URL: http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam12-scanmatching-mini.pdf (visited on 12/07/2021).

Sources (cont.)

[9] Ankit A. Ravankar et al. "Algorithms and a Framework for Indoor Robot Mapping in a Noisy Environment Using Clustering in Spatial and Hough Domains". In: *International Journal of Advanced Robotic Systems* 12.3 (2015), p. 27. DOI: 10.5772/59992. eprint: https://doi.org/10.5772/59992. URL: https://doi.org/10.5772/59992.

[10] "Scan Matching and SLAM for Mobile Robot in Indoor Environment". PhD thesis. 2016.

[11] Cyrill Stachniss. *Robot Mapping: Scan-Matching in 5 Minutes*. URL: http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam12-scanmatching-mini.pdf (visited on 12/07/2021).

Sources (cont.)

[12] Sebastian Thrun. "Robotic Mapping: A Survey". In: (Mar. 2002).

[13] Abby Yao. *Teaching Robots Presence: What You Need to Know About SLAM*. 2017. URL: https://blog.cometlabs.io/teaching-robots-presence-what-you-need-to-know-about-slam-9bf0ca037553 (visited on 12/07/2021).