Skoltech

Skolkovo Institute of Science and Technology

L15: Mapping

Perception in Robotics

Prof. Gonzalo Ferrer,

Skoltech, 15 March 2021

Mapping: a probabilistic perspective

We discussed long time ago, the probabilistic interpretation of estimating a map:

$$p(M \mid X, Z)$$

Observations Z are given and the trajectory X is also given by other means or they have been previously estimated, for instance after solving SLAM.

So far we have worked considering these maps a set of landmarks

$$M = \{l_1, l_2, \ldots, l_N\}$$

This was the introductory step, but there are other map representations that fulfill the belief of a map definition.

However, we might be interested in other map representation that provide more information of the environments, for example, for robot navigation.

Mapping representations

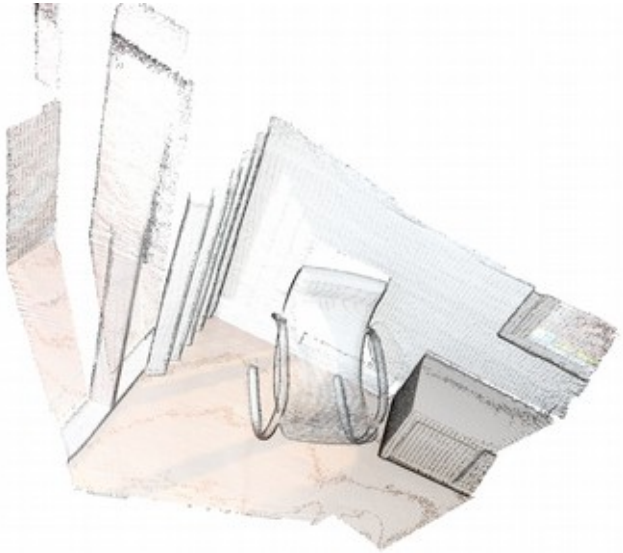
Explicit maps:

- Lines
- Corners
- Surfaces
- Mesh

Implicit maps:

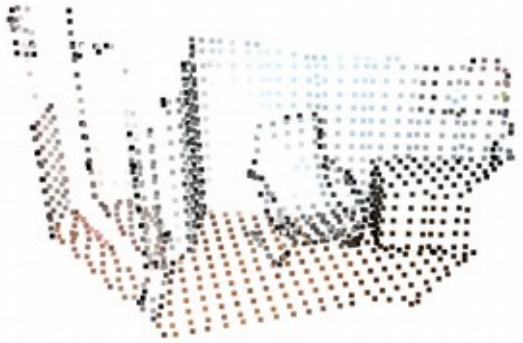
- Point Clouds
- Grid
- Surfels
- Voxels
- Octree
- Signed Distance Function (SDF)

Mapping representations



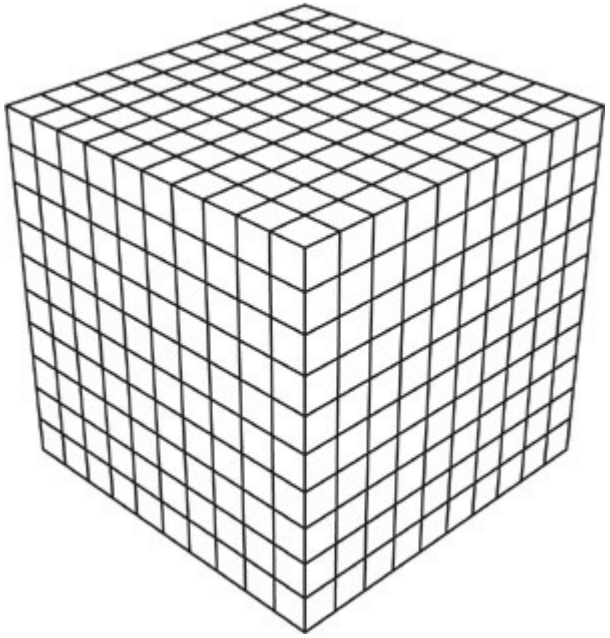
Point Clouds.

Visualization generated with open3d

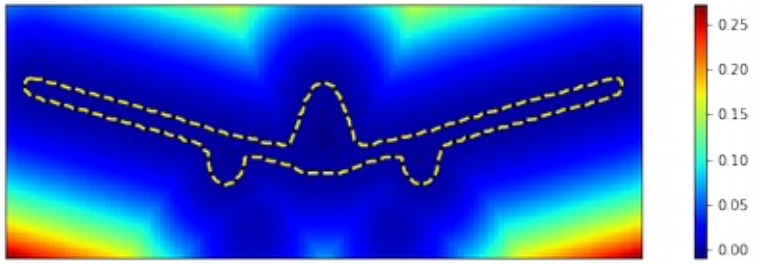


Voxel. The same point cloud is divided in equal cells and what is shown is the average of all points in the cell.

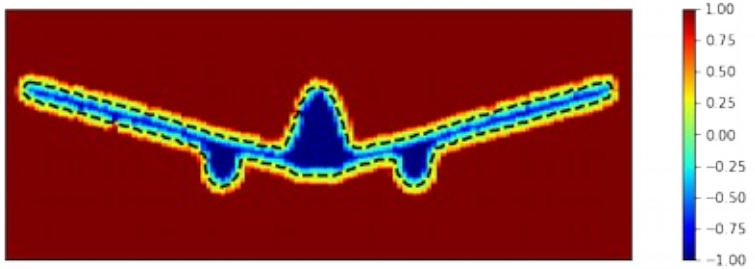
Mapping representations



3D Grid



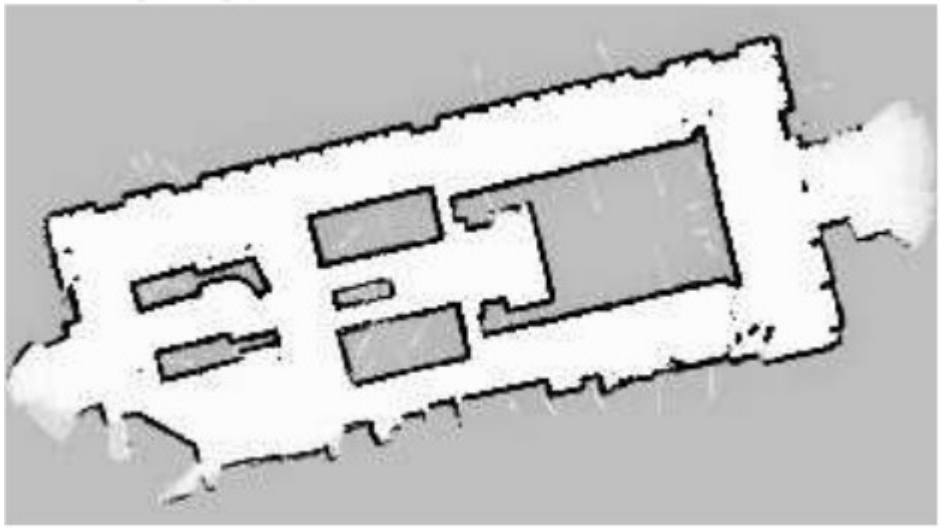
Signed Distance Function (SDF)



Truncated SDF (TSDF)

Occupancy grid mapping (ProbRob C9)

The occupancy grid mapping (OGM) is a very popular technique to represent environments.



In this example (taken from ProbRob), grid values of 0 encode free space and values of 1 encode occupied cell, meaning there is a wall, an object, something that the range sensor is perceiving. Binary state.

Grid mapping, state space

Let’s imagine a small example of 6x5 cells. Here, the maps can be defined as the set of values of each cell, being either occupied or free:

$$m = \{m_i\}_{6\times 5}$$

The combination of possible maps, that is the size of the state space, is huge:

- For a grid map of 6x5 there are 2^{30}, approximately 1000 M
- For a grid map of 100x100, the number is 2^{10 000} → intractable

We need some approximations:

1) Cells are independent: $p(m|X,Z) = \prod_i p(m_i|X,Z)$

2) Inverse model $p(m|z_t)$

We will show later how this 2 approximations will help on calculating the belief of the map.

OCM: Probability of Cells as Log Odds

For each cell, there is a binary state and its corresponding probability of is:

$$m_i = \begin{cases} 0 & \text{free} \\ 1 & \text{occupied} \end{cases} \qquad p(m_i) = \begin{cases} 0 & \text{free} \\ 1 & \text{occupied} \\ 0.5 & \text{no knowledge} \end{cases}$$

An alternative of using PDF when the state is binary is Log odds (ProbRob 4.2)

$$\frac{p(x)}{p(\bar{x})} = \frac{p(x)}{1 - p(x)} \quad \Longrightarrow \quad l(x) = \log \frac{p(x)}{1 - p(x)}$$

The random variable x is in [0,1] (binary), while r.v. $l \in (-\infty, \infty)$

OCM Algorithm

The log odds operation has a well defined inverse

$$p(x) = 1 - \frac{1}{1 + \exp(l(x))}$$

From here, we can define a an equivalent of a Binary Bayes filter for cells (ProbRob 9.2 and notes). The final result for each cell is:

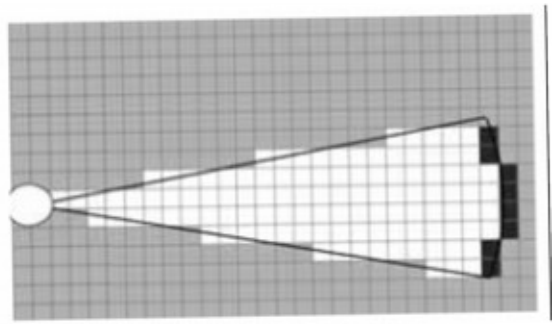
$$l_{t,i} = l(m_i|X, Z) = l_{t-1,i} + \text{inverse\_sensor\_model}(m_i, x_t, z_t) - l_o$$

OCM: the Inverse Sensor Model

The inverse sensor model returns ‘occupied’ or ‘free’ for each cell after ray-tracing over the cells for each of the observations.

For undefined result, returns the prior result.

Example:



Modern Mapping approaches

The OCM algorithm is a recursive filter that calculates and updates the values of each cell with each new observation. This is a probabilistic approach, usually considering a noisy sensor.

Other modern mapping approaches, follow a different strategy: Instead of calculating the posterior belief of the map, the process is simplified by assuming perfect localization and marginalizing poses:

$$p(m|z_{1:t}, u_{1:t}) = \int_{x_t} p(m, x_t|z_{1:t}, u_{1:t}) dx_t$$

$$= \int_{x_t} p(m|x_t, z_{1:t}, u_{1:t}) p(x_t|z_{1:t}, u_{1:t}) dx_t$$

$$\simeq \int_{x_t} p(m|x_t, z_{1:t}, u_{1:t}) \delta(x_t|z_{1:t}, u_{1:t}) dx_t$$

Modern Mapping with quality sensors

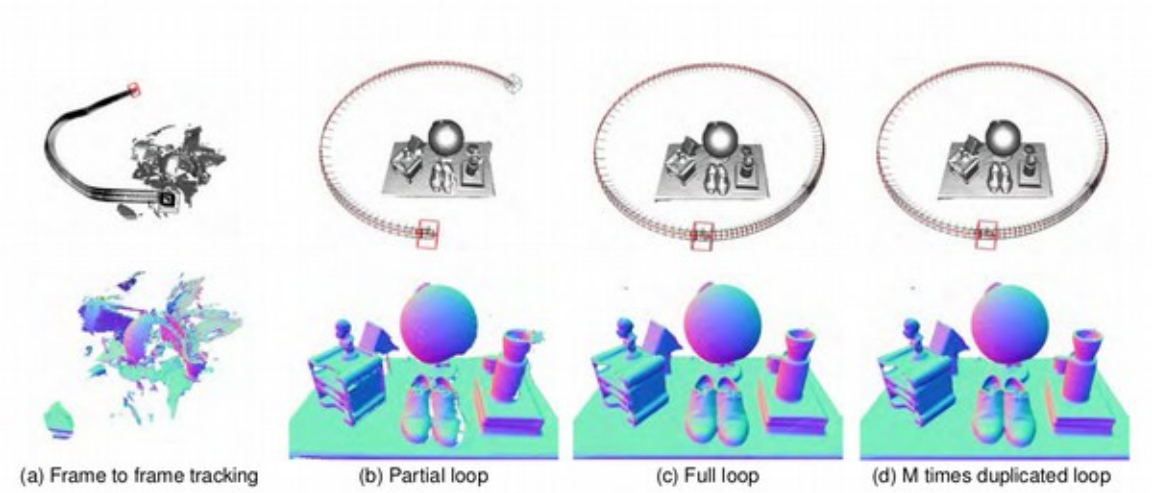
A different view to the probabilistic approach, is that we directly update the map, and this process is only possible when the sensors and the current maps are of high precision. Camera or sensor tracking.

The operation taking place is a Scan-2-Map or Map-2-Map. We discussed in L14, the alignment of point clouds is a particular case of this alignment.

Is then SLAM necessary? Yes, some of these approaches consider the effect of loop closure, to compensate for drift effects.

Kinect Fusion (Newcombe 2011)

They propose to update a detailed TSDF of the environment, using depth from kinect (dense point cloud).



The limitations are on the update of the TSDF, executed by GPU and the size of the environment, determined by the memory in the graphic card.

Newcombe, et al. ``Real-Time Dense Surface Mapping and Tracking'', ISMAR 2011

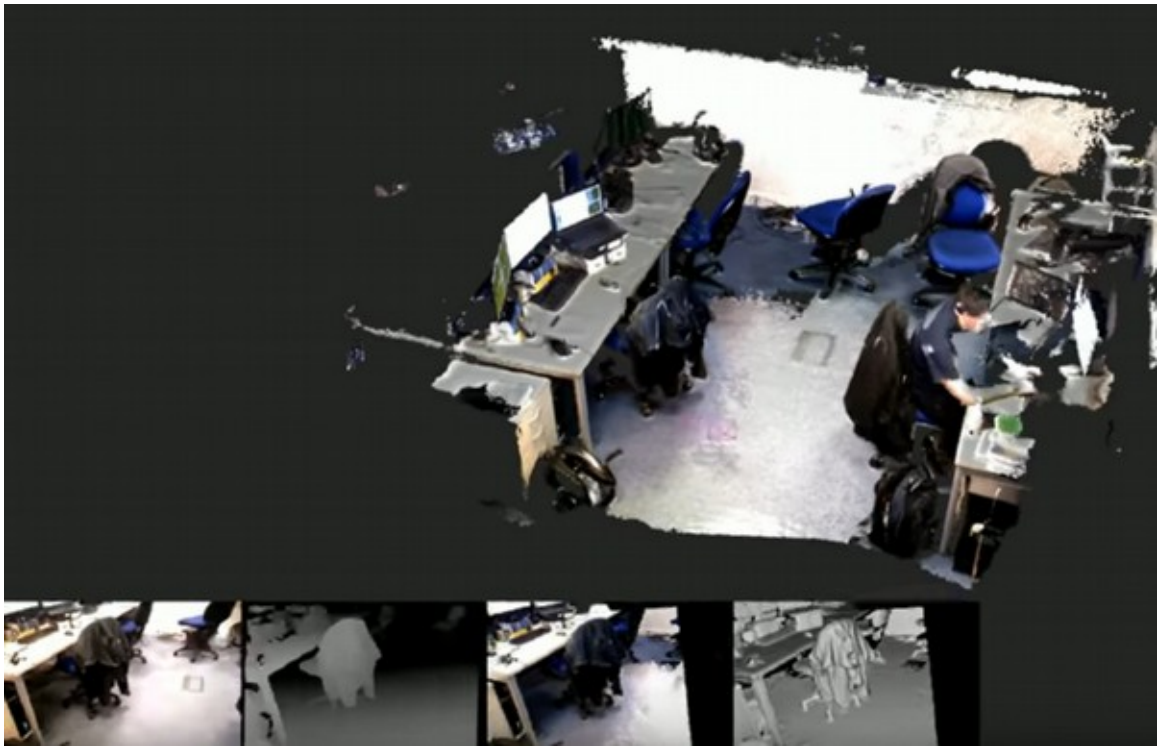
Elastic Fusion (Wheelan 2015)

Surfel-based map. Here they also use color (RGB) and track camera poses w.r.t the map of surfels.

The map is updated with fusion of surfels, but still, they marginalize poses.

Loop closure is done by non-rigid deformation of the map to compensate for possible drift.

Without pose graph.



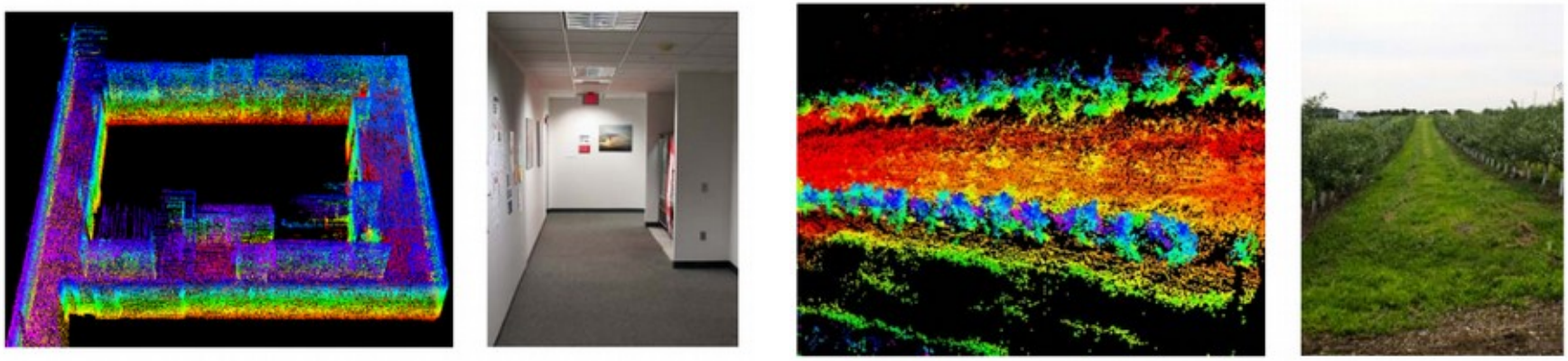
https://www.youtube.com/watch?v=XySrhZpODYs

Whelan, Thomas, et al. "ElasticFusion: Dense SLAM without a pose graph." Robotics: Science and Systems, 2015.

LOAM (Zhang 2015)

Map representation: voxel with sub-division of regions for efficiency.

The Alignment is done in two steps, one high frequency alignment for odometry and a second less frequent doing map-to-scan registration in real time.



Some examples of LOAM in indoor and outdoor enviroments.

Zhang, Ji, and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time". RSS 2015