Robot Autonomy

Graph SLAM - A Least Squares Approach

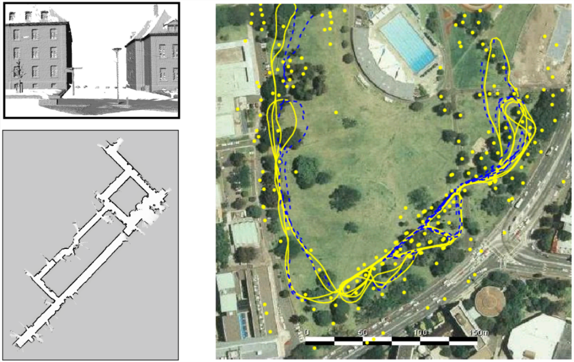
Dan M. Novischi

dan_marius.novischi@upb.ro

University POLITEHNICA of Bucharest
Faculty of Automatic Control And Computers

Map Representations

Features vs Metric



Feature Maps

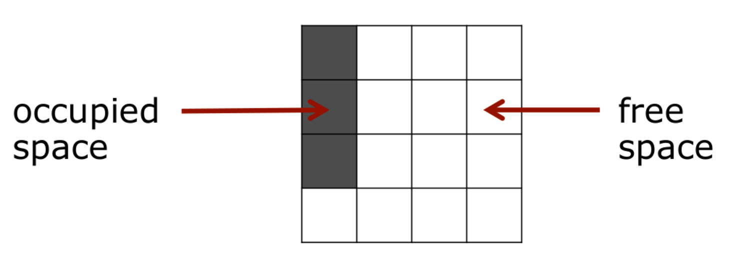
- Compact representation
- Memory efficient
- Needs a very good feature detector
- Must deal with data-association problems

Grid Maps

- Discretize the world into cells
- Each cell is either occupied or free space
- Non-parametric model
- Doesn’t rely on detecting features
- Requires substantial memory resources

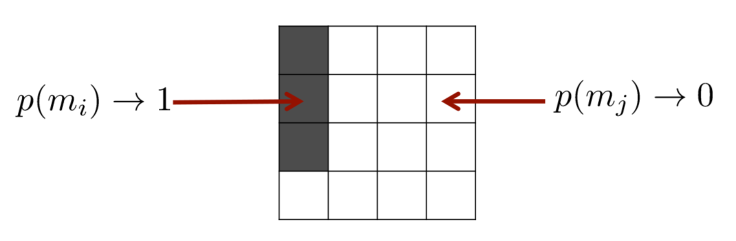
Grid Map Assumption 1

- A cell is completely free or occupied



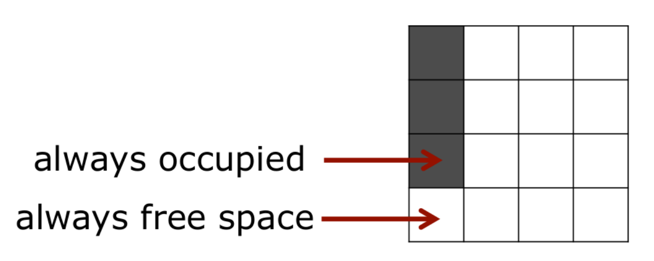
Occupancy Probabilistic Representation

- Each cell is associated as binary random variable
- The probability shows the belief of the cell being occupied or free



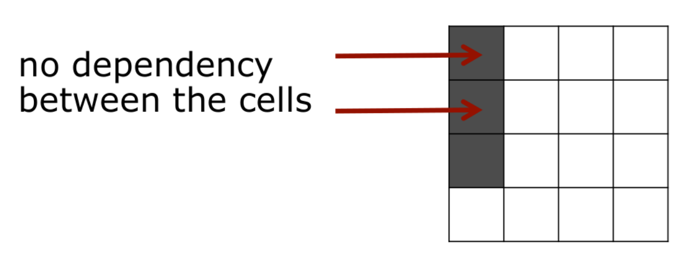
Grid Map Assumption 2

- The world is **static**



Grid Map Assumption 3

- The binary random variables are independent



Occupancy Map Representation

- The probability distribution of the map is given by the product of the probability over the cells

$$\underset{\uparrow \atop \text{map}}{p(m)} = \prod_i \underset{\uparrow \atop \text{cell}}{p(m_i)}$$

Estimating a Map from Data

- Estimating the map given the sensor data $z_{1:t}$ and the poses $x_{1:t}$ translates to:

$$p(m \mid z_{1:t}, x_{1:t}) \quad = \quad \prod_i p(m_i \mid z_{1:t}, x_{1:t})$$

binary random variable

Least Squares Overview

- Generic approach to many optimization related problems (including ML)
- Computes solutions for over-determined systems
- More equation than unknowns
- Seeks to minimize the sum of squared errors
- Deeply related to Linear Regression in ML

Least Squares Problem Setup

- Given an system described by a set of n observation functions $\{f_i(x)_{i=1:n}\}$
- Let:
  - $\mathbf{X}$ be the state vector
  - $\mathbf{Z}_i$ be a measurement of the state $\mathbf{X}$
  - $\hat{\mathbf{Z}}_i = f_i(\mathbf{X})$ be a function that maps $\mathbf{X}$ to a measurement $\hat{\mathbf{Z}}_i$
- Given: n noisy measurements $\mathbf{Z}_{1:n}$ about the state $\mathbf{X}$
- Goal: estimate state $\mathbf{X}$ which best explain the measurements $\mathbf{Z}_{1:n}$

Least Squares Error Function

- Error $\mathbf{e}_i$ is usually the difference between the predicted and the actual measurement

$$\mathbf{e}_i(x) = \mathbf{z}_i - \hat{\mathbf{z}}_i = \mathbf{z}_i - f_i(\mathbf{x})$$

- We assume the error is normally distributed with zero mean
- Gaussian error has the information matrix $\mathbf{\Omega}_i$
- Squared error of a measurement depends only on the state and is a scalar

$$e_i(\mathbf{x}) = \mathbf{e}_i(\mathbf{x})^T \, \mathbf{\Omega}_i \, \mathbf{e}_i(\mathbf{x})$$

Least Squares Goal

- Finding the x^* entails minimizing the error given all measurements

$$
\begin{aligned}
x^* &= \underset{x}{\operatorname{argmin}} F(\mathbf{x}) \\
&= \underset{x}{\operatorname{argmin}} \sum_i e_i(\mathbf{x}) \\
&= \underset{x}{\operatorname{argmin}} \sum_i \mathbf{e}_i^T(\mathbf{x})\ \boldsymbol{\Omega}_i\ \mathbf{e}_i(\mathbf{x})
\end{aligned}
$$

- Usual approach is to find the derivative nulls (zeros).
- Complex and no closed form solution

Least Squares Numerical Solution Assumptions

- We can construct a good initial guess
- Error functions are smooth in the neighborhood of the (global) minima
- So, we can iteratively solve the problem by computing local liniarizations at each step

Least Squares Numerical Solution Steps

- Liniarize the error terms around the current solution (the starting point is our initial guess)
- Compute the first order derivative of the error function
- Set it to zero and solve the linear system (to obtain a better solution)
- Iterate until convergence

Error Function Linearization

- Linearizing via Taylor series expansion gives:

$$\mathbf{e}_i(\mathbf{x} + \mathbf{\Delta x}) \simeq \mathbf{e}_i(\mathbf{x}) + \mathbf{J}_i(\mathbf{x})\mathbf{\Delta x}$$

- where the Jacobian is given by:

$$\mathbf{J}_f = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}$$

Squared Error Minimization

- We can fix **x** and do the minimization in increments **Δx**
- Replacing the Taylor expansion in the squared error terms yields:

$$
\begin{aligned}
\mathbf{e}_i(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x}) &= \mathbf{e}_i^T(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x})\ \boldsymbol{\Omega}_i\ \mathbf{e}_i(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x}) \\
&\simeq (\mathbf{e}_i+\mathbf{J}_i\boldsymbol{\Delta}\mathbf{x})^T\ \boldsymbol{\Omega}_i\ (\mathbf{e}_i+\mathbf{J}_i\boldsymbol{\Delta}\mathbf{x}) \\
&\quad \mathbf{e}_i^T\ \boldsymbol{\Omega}_i\ \mathbf{e}_i + \\
&= \mathbf{e}_i^T\ \boldsymbol{\Omega}_i\ \mathbf{J}_i\boldsymbol{\Delta}\mathbf{x}\ +\ \boldsymbol{\Delta}\mathbf{x}^T\ \mathbf{J}_i^T\ \mathbf{e}\ \boldsymbol{\Omega}_i + \\
&\quad \boldsymbol{\Delta}\mathbf{x}^T\ \mathbf{J}_i^T\ \boldsymbol{\Omega}_i\ \mathbf{J}_i\ \boldsymbol{\Delta}\mathbf{x}
\end{aligned}
$$

- Manipulating the equation result in $\Delta x^* = -H^{-1}b$ with $H = \sum_i \mathbf{J}_i^T\ \boldsymbol{\Omega}_i\ \mathbf{J}_i$ and $b^T = \sum_i \mathbf{e}_i^T\ \boldsymbol{\Omega}_i\ \mathbf{J}_i$

Long Story Short

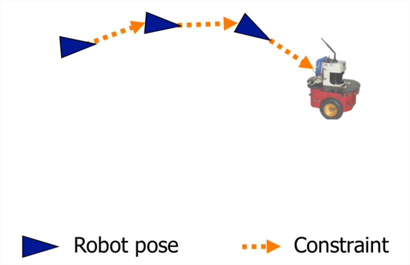
- Liniarize around x and compute for each measurement:

$$\mathbf{e}_i(\mathbf{x} + \mathbf{\Delta x}) \simeq \mathbf{e}_i(\mathbf{x}) + \mathbf{J}_i(\mathbf{x})\mathbf{\Delta x}$$

- Compute linear system terms $H = \sum_i \mathbf{J}_i^T \ \mathbf{\Omega}_i \ \mathbf{J}_i$ and $b^T = \sum_i \mathbf{e}_i^T \ \mathbf{\Omega}_i \ \mathbf{J}_i$
- Solve the linear system $\Delta x^* = -H^{-1}b$
- Update state $x = x + \Delta x^*$
- Iterate until convergence

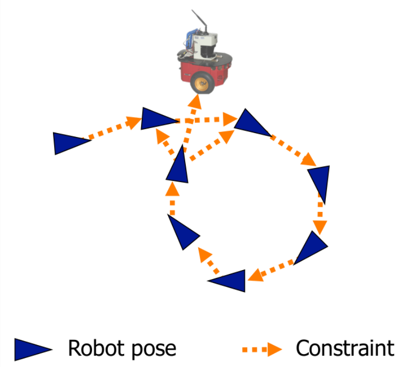
Graph SLAM - Motion

- Constraints connect successive poses while the robot is moving
- Constraints are inherently uncertain



Graph SLAM - Observation

- Observing previous location generates constraints between non-successive poses

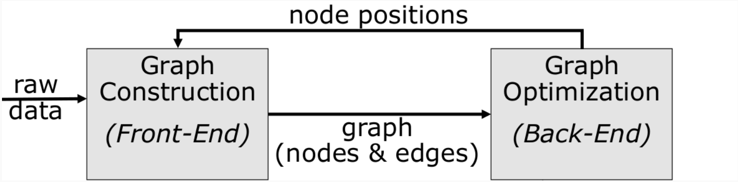


Graph SLAM Idea

- Represent the SLAM problem as a graph
- Every node corresponds to a pose at which we took a measurement
- Every edge (between two nodes) represents a spatial constraint
- Graph SLAM entails building the graph an then minimizing the error introduced by the constraints

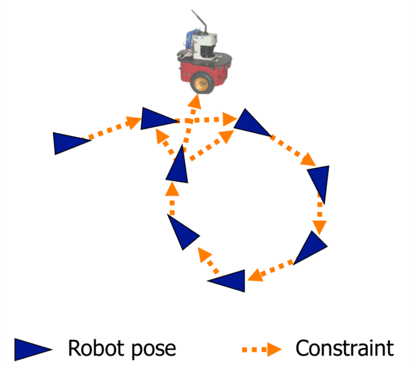
The Overall System

- Interplay between a front-end and a back-end
- Front-end successively builds the graph
- Back-end successively optimizes the graph



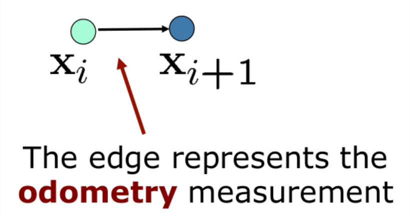
The Graph

- Consists of n nodes $x_{1:n}$
- Each x_i is the pose of a robot at time t_i
- A constraint/edge exists between nodes x_i and x_j if ...



Robot movement constraints

- ...the robot moves from x_i to $x_i + 1$
- Edge corresponds to the odometry measurement



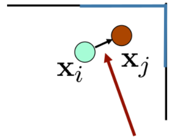
Robot observation Constraints

- ...the robot observes the same part of the environment from x_i and from x_j

Measurement from $\mathbf{x}_i$ Measurement from $\mathbf{x}_j$

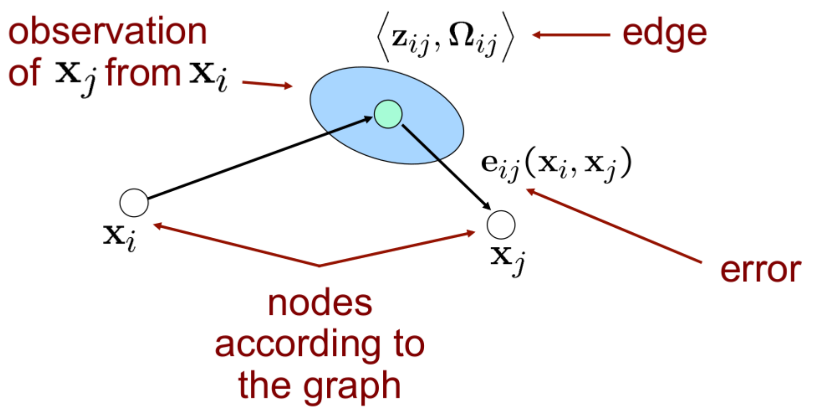
Robot Observation Constraints

- ...the robot observes the same part of the environment from x_i and from x_j
- Construct a virtual measurement about the position of x_j seen from x_i



Edge represents the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$ based on the **observation**

The Pose Graph



Optimize via Least Squares

- Error looks according to the least squares formulation

$$x^* = \operatorname*{argmin}_x \sum_{ij} e_{ij}^T(x_i, x_j)\Omega_{ij}e_{ij}(x_i, x_j)$$
$$= \operatorname*{argmin}_x \sum_{k} e_k^T(x)\Omega_k e_k(x)$$

- We now have a way to recover the correct poses
- Then mapping with known poses becomes a very easy task

Summary

- Occupancy grid maps discretize the space into independent cells
- Each cell is a binary random variable estimating if the cell is occupied
- Least Squares: technique to minimize squared error
- We can represent the SLAM problem as a graph
- Back-end can be effectively implemented via least squares
- More information in [1, 2]

References I

G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard.
A tutorial on graph-based slam.
IEEE Intelligent Transportation Systems Magazine, 2(4):31–43, 2010.

S. Thrun.
Probabilistic robotics.
Communications of the ACM, 45(3):52–57, 2002.