Università degli Studi di Padova

IAS-Lab
Intelligent Autonomous Systems Laboratory

A Very Brief Introduction to the Localization and SLAM Problems
Part 3

Alberto Pretto

Thanks to Wolfram Burgard, Giorgio Grisetti, Davide Scaramuzza and Cyrill Stachniss for some slides!

Probabilistic SLAM

Estimate the robot's path and the map

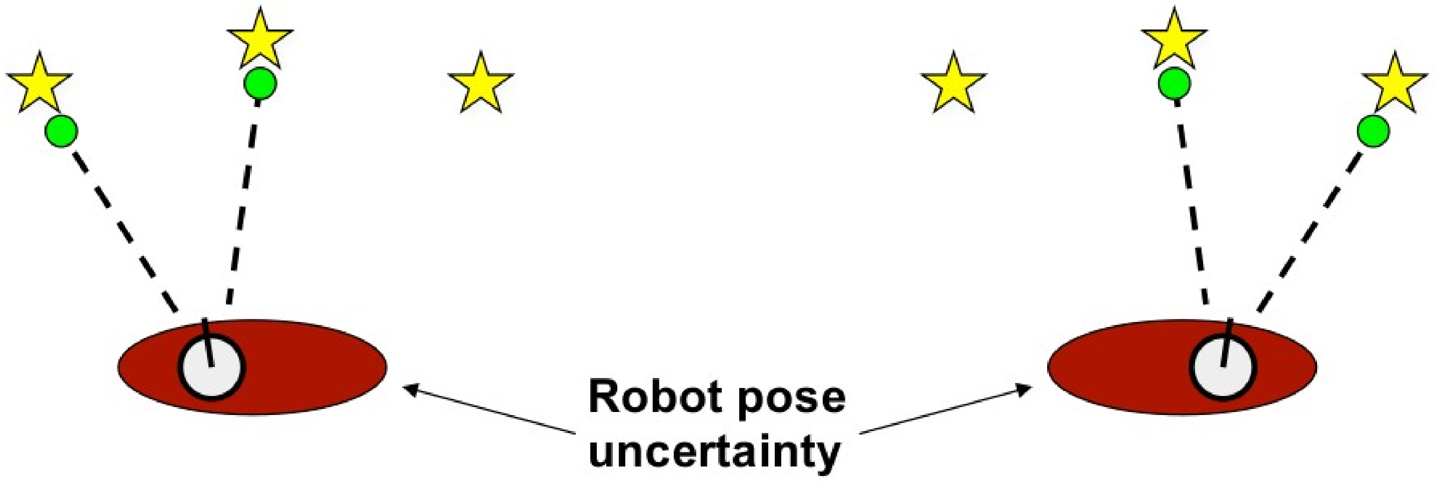
$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution path map given observations controls

Why is SLAM a Hard Problem?

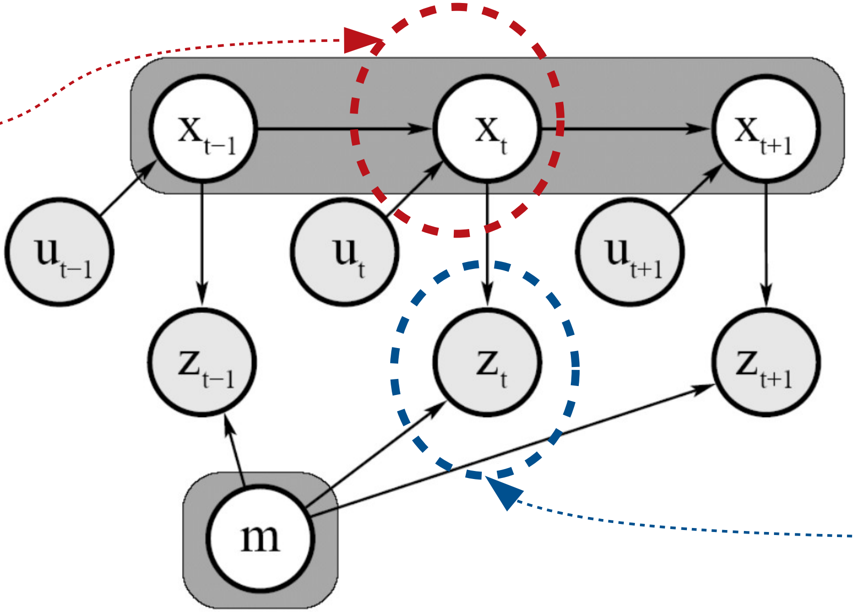
The mapping between observations and the map is unknown

Picking **wrong data associations** can have catastrophic consequences



SLAM Graphical Model

Motion model

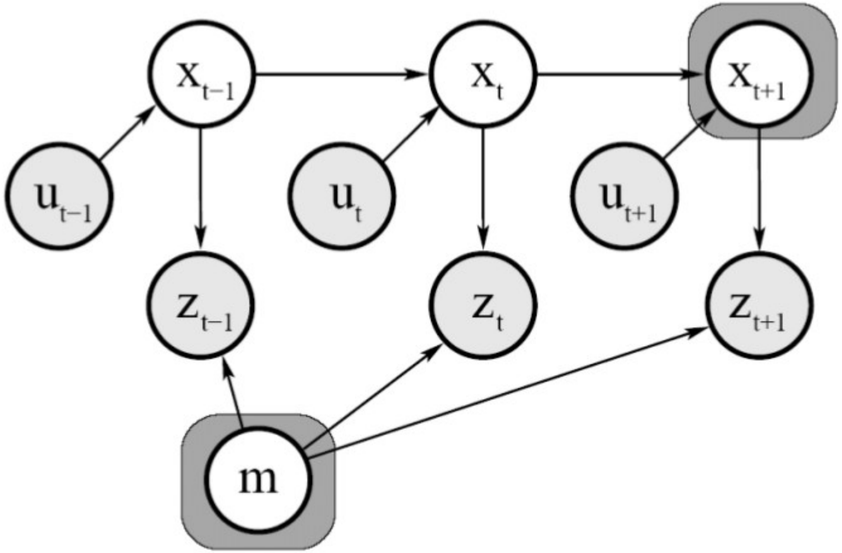


Observation model

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

Often called **Full** SLAM

Online SLAM Graphical Model



Main Paradigms

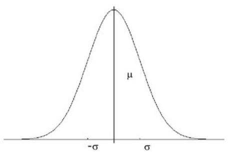
Kalman filter

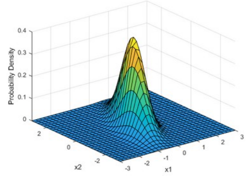
Particle filter

Graph-based

SLAM with Kalman Filter

If we assume that all the involved PDFs are Gaussians, we can solve the SLAM problem with the **Kalman Filter**





$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

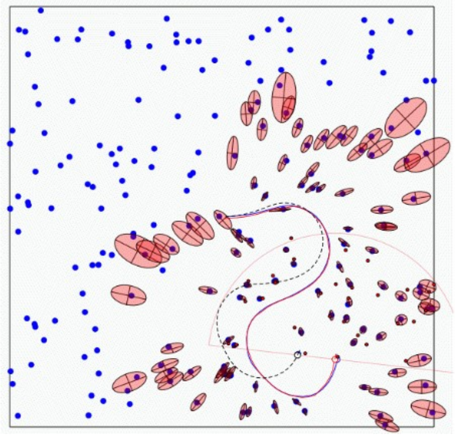
$$p(x_t \mid u_t, x_{t-1})$$ Motion model

$$z_t = C_t x_t + \delta_t$$

$$p(z_t \mid x_t)$$ Sensor model

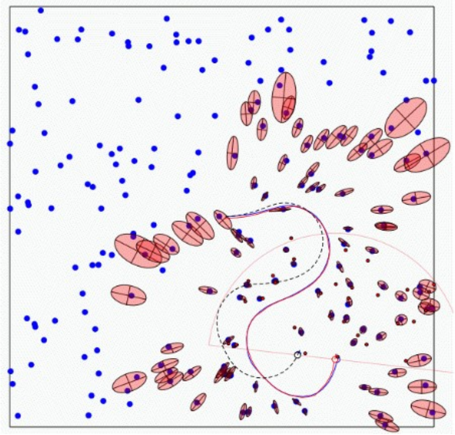
Landmark-Based SLAM with KF

Landmark-based online SLAM: add landmark positions $l_i = (x_i, y_i)$ directly to the basic localization state, and update them.



Landmark-Based SLAM with KF

Landmark-based online SLAM: add landmark positions $l_i = (x_i, y_i)$ directly to the basic localization state, and update them.



$$
\left\langle
\begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix},
\begin{pmatrix}
\sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\
\sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\
\sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\
\sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\
\sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2
\end{pmatrix}
\right\rangle
$$

Landmark-Based SLAM with KF

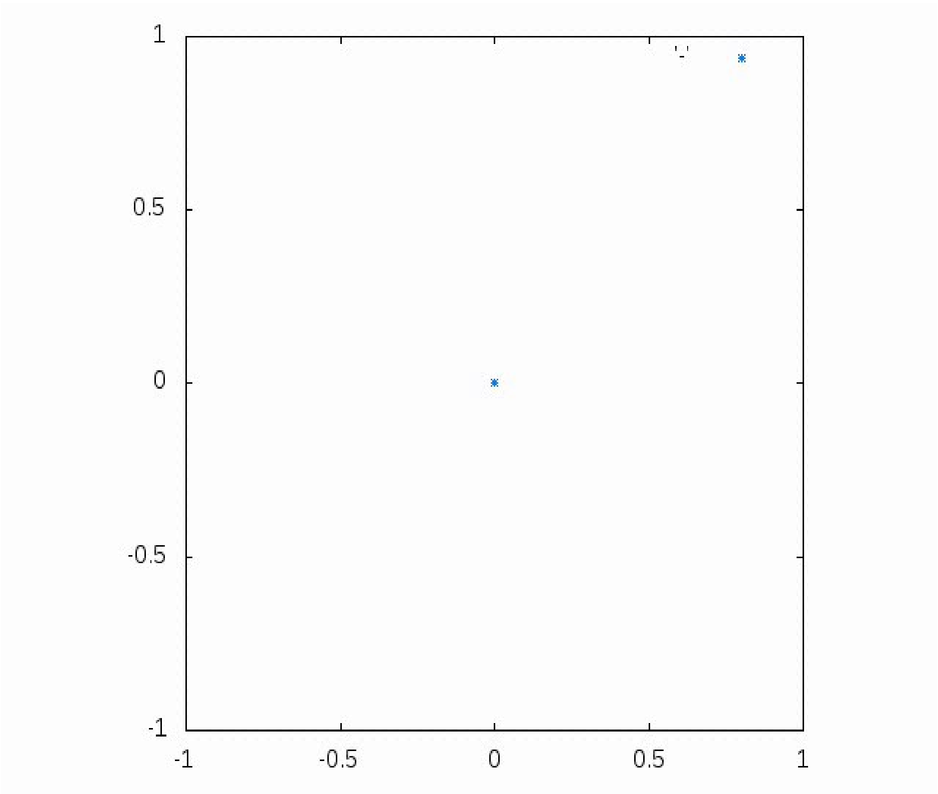
- Map with *N* landmarks: *(3+2N)*-dimensional Gaussian density (planar motion case)
- Can handle only up to a few hundred landmarks

$$
\left\langle
\begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix},
\begin{pmatrix}
\sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\
\sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\
\sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\
\sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\
\sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2
\end{pmatrix}
\right\rangle
$$

- The state size is not fixed: every time a new landmark j is detected, the state should be expanded to insert the new position l_j
- **A perfect data association is essential**

Landmark-Based SLAM with KF

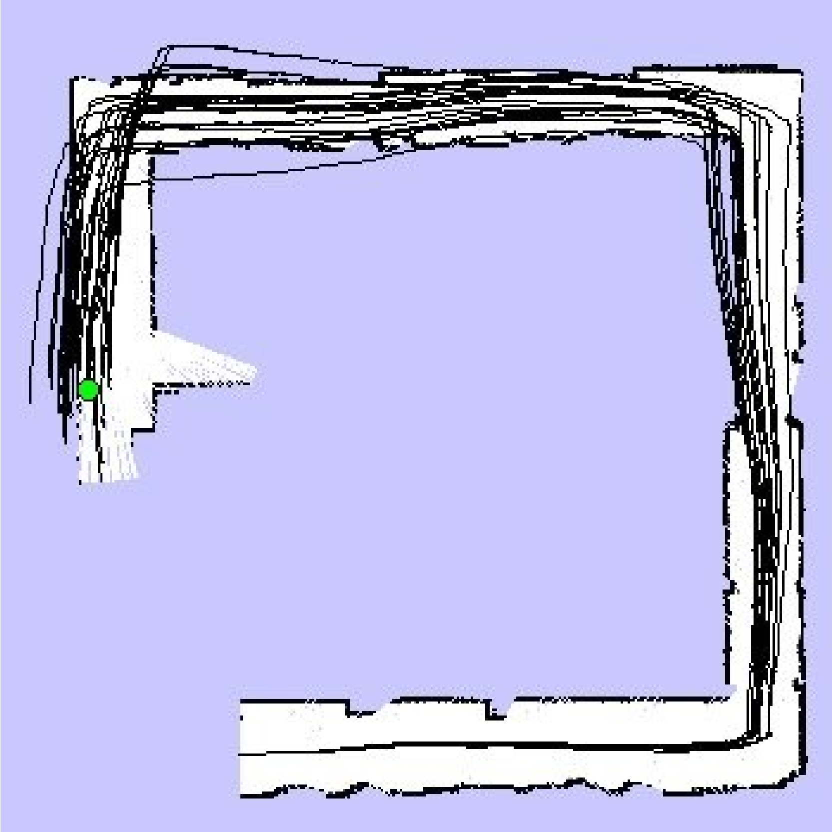
Landmark-Based SLAM with KF



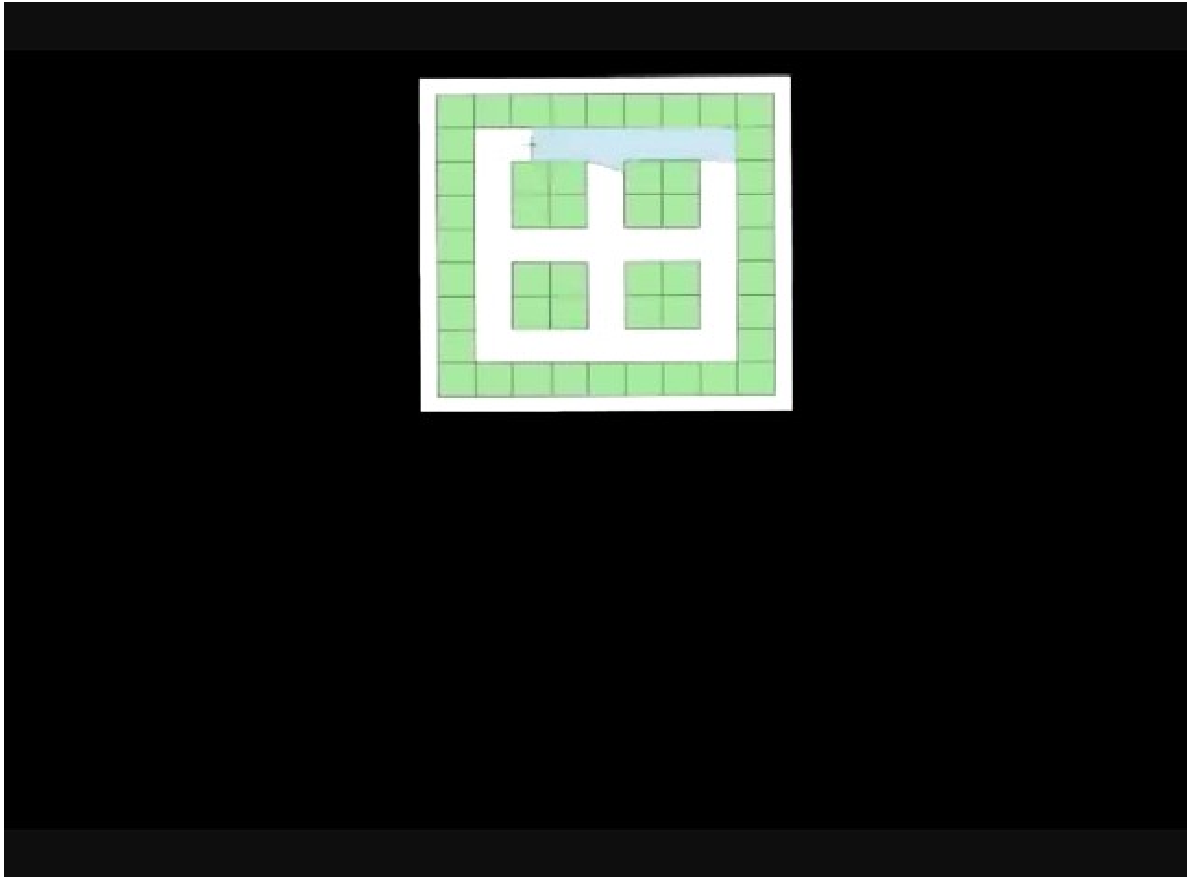
SLAM with Particle Filter

Insight: Particles are **full trajectories hypotheses**

- Update each trajectory with the more recent motion, and "weight" each trajectory with the more recent sensor reading.
- Trajectories with high weight (likelihood) will be easily duplicated, vice versa will be easily removed.

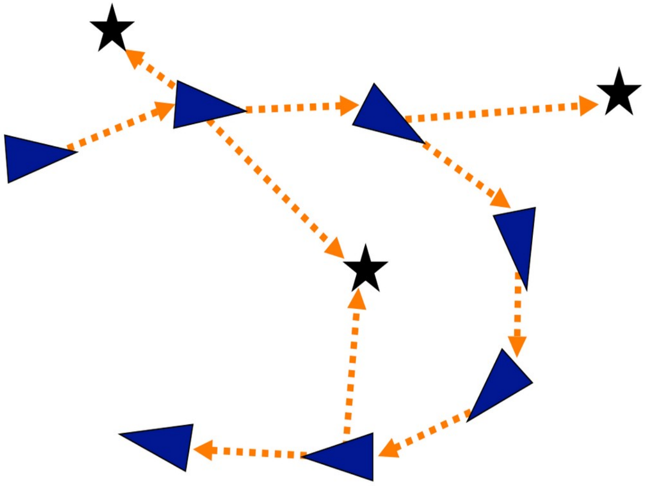


SLAM with Particle Filter



Graph-based SLAM

- Full SLAM solution
- Landmarks and robot locations as nodes of a graph
- Pair of consecutive locations x_i and x_{i+1} are linked by an edge
- If the landmark m_j is observed by location x_i, they are linked by an edge
- Each edge represents a conditional dependency, i.e. **a sort of constraint**



Graph-based SLAM: Derivation

We provide here an intuitive (**not formal/not comprehensive**) derivation of a basic **landmark based** graph SLAM solution

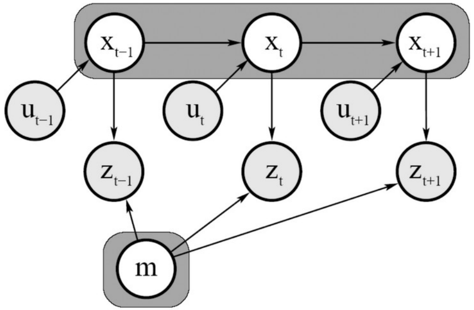
$$(Definition : p(x_{1:t}) := p(x_1, x_2, \ldots, x_t))$$

SLAM target density ➡ $p(x_{1:t}, m | z_{1:n}, u_{1:t-1})$

Bayes rule $= \dfrac{p(z_{1:n} | x_{1:t}, m, u_{1:t-1}) \; p(x_{1:t}, m | u_{1:t-1})}{p(z_{1:n} | u_{1:t-1})}$

Normalization $= \eta \; p(z_{1:n} | x_{1:t}, m, u_{1:t-1}) \; p(x_{1:t}, m | u_{1:t-1})$

Graph-based SLAM: Derivation



$$p(z_{1:n} | x_{1:t}, m, u_{1:t-1})$$

Markov $$= p(z_1 | x_{1:t}, m, u_{1:t-1}) \; p(z_2 | x_{1:t}, m, u_{1:t-1}) \; \ldots \; p(z_n | x_t, m, u_{1:t-1})$$

Markov $$= p(z_1 | x_{i_{z_1}}, m) \; p(z_2 | x_{i_{z_2}}, m) \; \ldots \; p(z_n | x_{i_{z_n}}, m)$$

The location where the observation z_1 has been collected

Sensor model: we know how to compute it!

Graph-based SLAM: Derivation

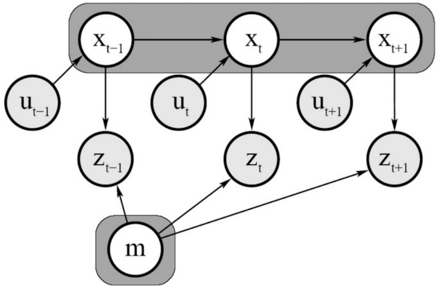
$$p(x_{1:t}, m | u_{1:t-1})$$

Chain rule $$= p(x_t | x_{1:t-1}, m, u_{1:t-1})\ p(x_{1:t-1}, m | u_{1:t-1})$$

Markov $$= p(x_t | x_{1:t-1}, u_{1:t-1})\ p(x_{1:t-1}, m | u_{1:\underline{t-2}})$$

Markov $$= p(x_t | \underline{x_{t-1}}, u_{t-1})\ p(x_{1:t-1}, m | u_{1:t-2})$$

Chain rule + Markov + Markov.... $$\cdots$$



$$= p(x_t | x_{t-1}, u_{t-1})\ p(x_{t-1} | x_{t-2}, u_{t-2})$$
$$p(x_{t-2} | x_{t-3}, u_{t-3})\ \ldots\ p(x_0)$$

Motion model: we know how to compute it!

Prior

Graph-based SLAM: Derivation

- Putting everything together:

$$p(x_{1:t}, m | z_{1:n}, u_{1:t-1})$$

$$= \eta \; p(z_{1:n} | x_{1:t}, m, u_{1:t-1}) \; p(x_{1:t}, m | u_{1:t-1})$$

$$= \eta \, p(z_1 | x_{i_{z_1}}, m) \; \ldots \; p(z_n | x_{i_{z_n}}, m) \; p(x_t | x_{t-1}, u_{t-1}) \; \ldots \; p(x_0)$$

- Computing a SLAM solution is equivalent to computing the state $x^*_{1:t}, m^*$, i.e. the configuration of the nodes, that maximizes

$$p(x_{1:t}, m | z_{1:n}, u_{1:t-1})$$

Graph-based SLAM: Derivation

In other words, solving a Graph-based SLAM problem is equivalent to a maximum likelihood approach to find:

$$x^*_{1:t}, m^* = argmax_{x_{1:t},m}\ p(x_{1:t}, m | z_{1:n}, u_{1:t-1})$$

This is equivalent to minimize the negative natural logarithm probability:

$$x^*_{1:t}, m^* = argmin_{x_{1:t},m}\ (-\ln (p(x_{1:t}, m | z_{1:n}, u_{1:t-1})))$$

Graph-based SLAM: Derivation

Remembering the previous derivation, and applying the logarithm of a product property ***ln(ab)=ln(a)+ln(b)***:

$$x_{1:t}^*, m^* = argmin_{x_{1:t},m} \left(-\ln\left(p(x_{1:t}, m | z_{1:n}, u_{1:t-1})\right)\right)$$

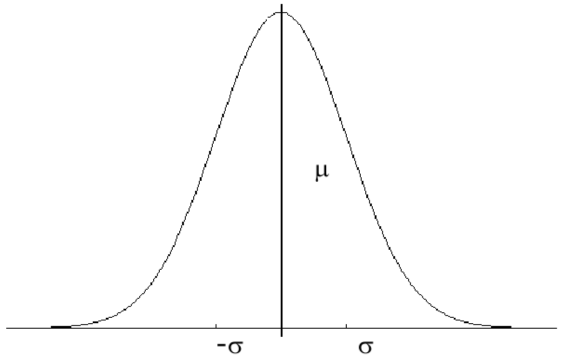
$$= argmin_{x_{1:t},m} \left(const - \sum_{i=1}^{n} \ln(p(z_i | x_{j_{z_i}}, m)) - \sum_{i=2}^{t} \ln(p(x_i | x_{i-1}, u_{i-1})) \right)$$

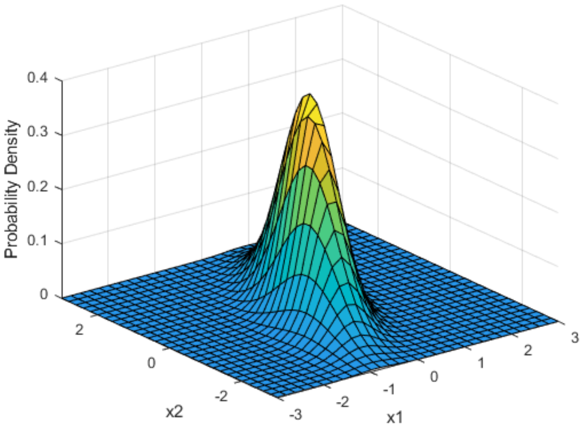
$$= argmin_{x_{1:t},m} \left(- \sum_{i=1}^{n} \ln(p(z_i | x_{j_{z_i}}, m)) - \sum_{i=2}^{t} \ln(p(x_i | x_{i-1}, u_{i-1})) \right)$$

Graph-based SLAM: Derivation

As done for the Kalman filter, we can assume to represent all densities by menas of (uni or multivariate) Gaussians:

$$X \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)}$$





Graph-based SLAM: Derivation

In the Gaussian domain, applying the natural logarithm property ***ln e^x = x***:

$$x^*_{1:t}, m^* = argmin_{x_{1:t},m} \left(-\ln\left(p(x_{1:t}, m | z_{1:n}, u_{1:t-1})\right)\right)$$

-ln(η e^{-x}) = const + x

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - \mu_{z,i})^T \Sigma_z^{-1}(z_i - \mu_{z,i}) + \frac{1}{2}\sum_{i=2}^{t}(x_i - \mu_{x,i})^T \Sigma_u^{-1}(x_i - \mu_{x,i})$$

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - h(x_{j_{z_i}}, m)))^T \Sigma_z^{-1}(z_i - h(x_{j_{z_i}}, m) + \frac{1}{2}\sum_{i=2}^{t}(x_i - g(x_{i-1}, u_{i-1}))^T \Sigma_u^{-1}(x_i - g(x_{i-1}, u_{i-1}))$$

Map the current state into an expected observation z_i

Using the action, map the current state into x_i

Graph-based SLAM: Derivation

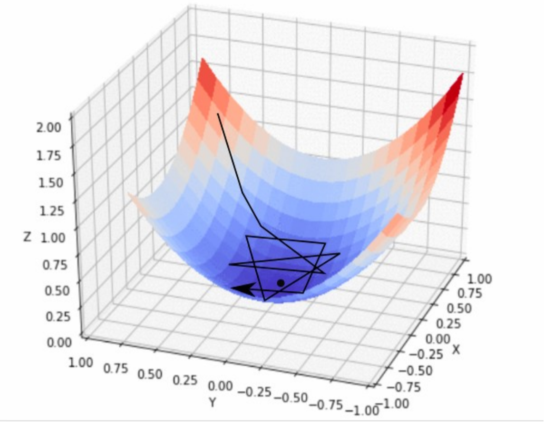
$$x_{1:t}^*, m^* = argmin_{x_{1:t},m} \left(-\ln\left(p(x_{1:t}, m | z_{1:n}, u_{1:t-1})\right)\right)$$

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - \mu_{z,i})^T \Sigma_z^{-1}(z_i - \mu_{z,i}) \;+\; \frac{1}{2}\sum_{i=2}^{t}(x_i - \mu_{x,i})^T \Sigma_u^{-1}(x_i - \mu_{x,i})$$

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - h(x_{j_{z_i}}, m)))^T \Sigma_z^{-1}(z_i - h(x_{j_{z_i}}, m) \;+\; \frac{1}{2}\sum_{i=2}^{t}(x_i - g(x_{i-1}, u_{i-1}))^T \Sigma_u^{-1}(x_i - g(x_{i-1}, u_{i-1}))$$

We reformulated our SLAM problem into a least squares approach

→ Noteworthy observation: **squares on residuals are due to Gaussian assumption**



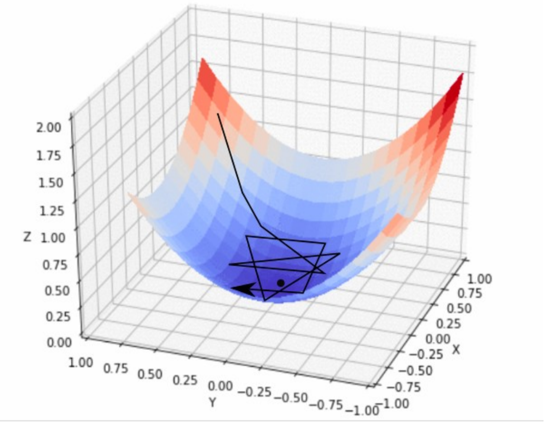
Graph-based SLAM: Derivation

$$x^*_{1:t}, m^* = argmin_{x_{1:t},m}\left(-\ln\left(p(x_{1:t}, m|z_{1:n}, u_{1:t-1})\right)\right)$$

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - \mu_{z,i})^T\Sigma_z^{-1}(z_i - \mu_{z,i}) + \frac{1}{2}\sum_{i=2}^{t}(x_i - \mu_{x,i})^T\Sigma_u^{-1}(x_i - \mu_{x,i})$$

$$= \frac{1}{2}\sum_{i=1}^{n}(z_i - h(x_{j_{z_i}}, m)))^T\Sigma_z^{-1}(z_i - h(x_{j_{z_i}}, m) + \frac{1}{2}\sum_{i=2}^{t}(x_i - g(x_{i-1}, u_{i-1}))^T\Sigma_u^{-1}(x_i - g(x_{i-1}, u_{i-1}))$$

It is a **weighted** least squares problem: less noisy sensors (intuitively: with a covariance matrix with smaller trace) will have more influence in computation of the cost function



Graph-based SLAM Solution

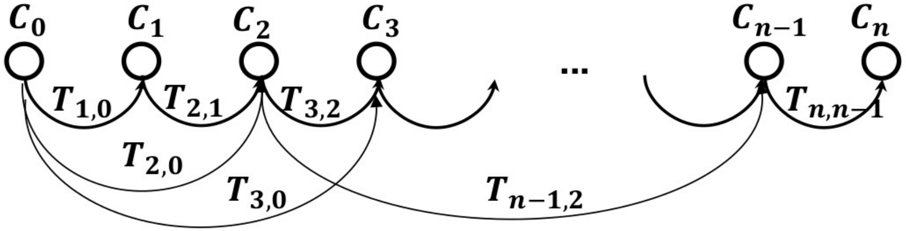
- If the map *m* is landmark based:

$$m := (m_1, \ldots, m_k)$$

  just solve the previous least square problem by means of standard optimization methods, like the **Gauss-Newton** or the **Levenberg-Marquardt** algorithms, by inferring both robot positions and landmark locations.
- What can be done if the map *m* is volumetric, e.g. grid-based, when the cardinality of the map is huge?

Graph-SLAM with a Pose Graph

- Include in the state just the robot path, **not** the map
- Solve SLAM by means of an optimization problem that takes into account consecutive transformations and loop closure detections given **both observations and actions**



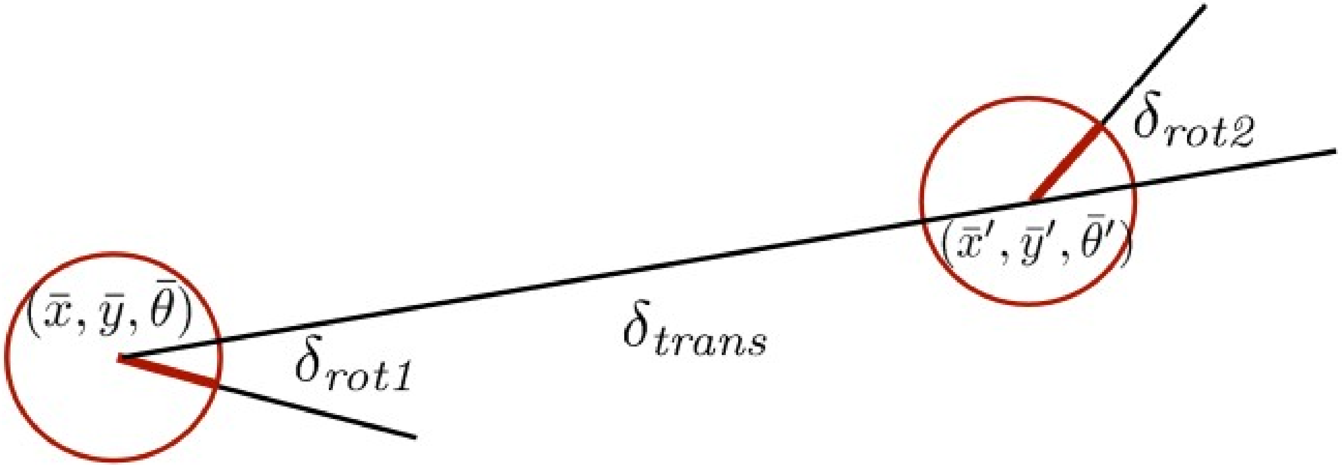
- Minimize:

$$E(\mathbb{T}) = \sum_i f(\mathbf{T}_i, \mathbf{T}_{i+1}, \mathbf{R}_i) + \sum_{i,j} f(\mathbf{T}_i, \mathbf{T}_j, \mathbf{T}_{ij}).$$

- Then, **just render the map given the robot positions**

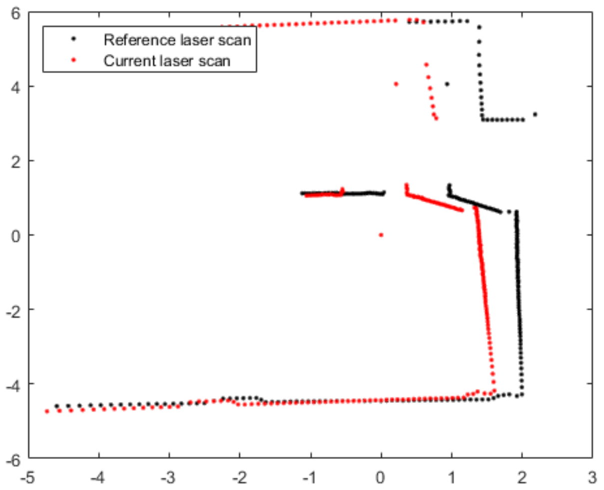
Pose Graph with Scan Matching

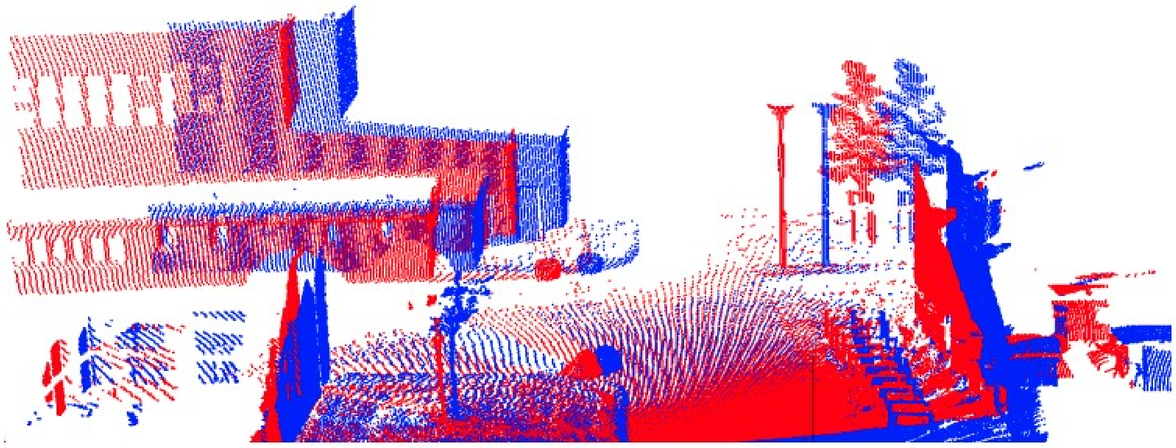
- For robot with 2D or 3D range finders (e.g., a LiDAR)
- As usual, use the odometry to get a prior of the robot motions between consecutive positions



Pose Graph with Scan Matching

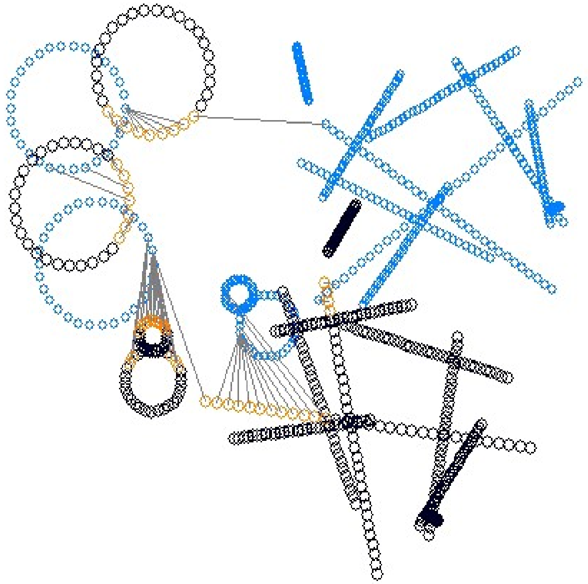
Incrementally align two scans (or a map to a scan) without revising the past/map.



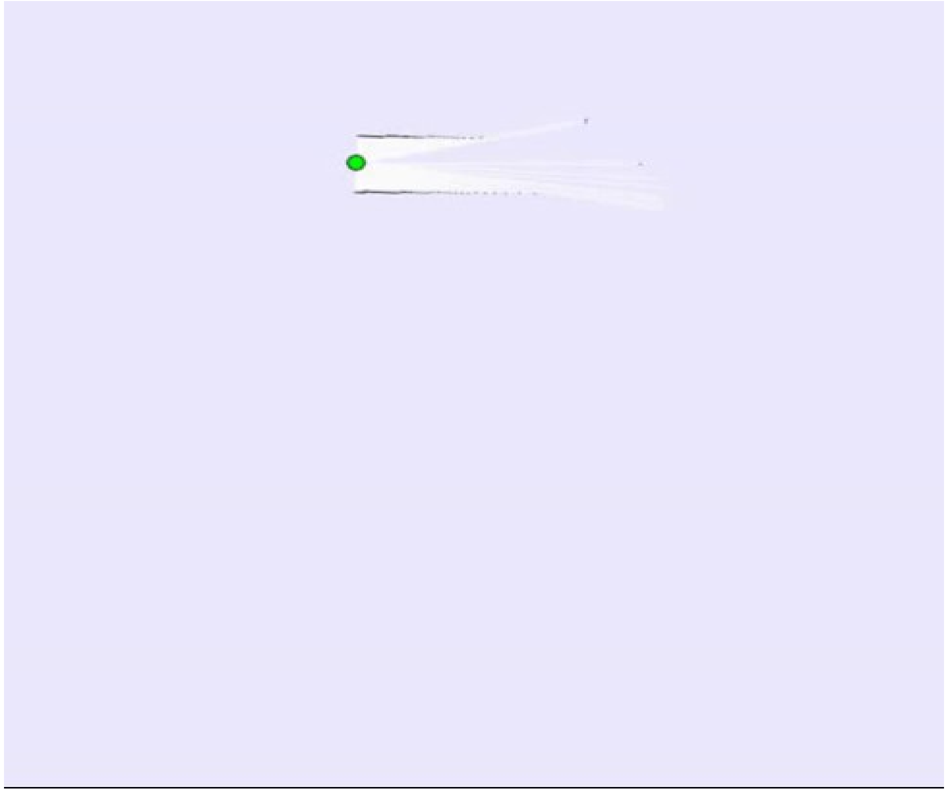


Pose Graph with Scan Matching

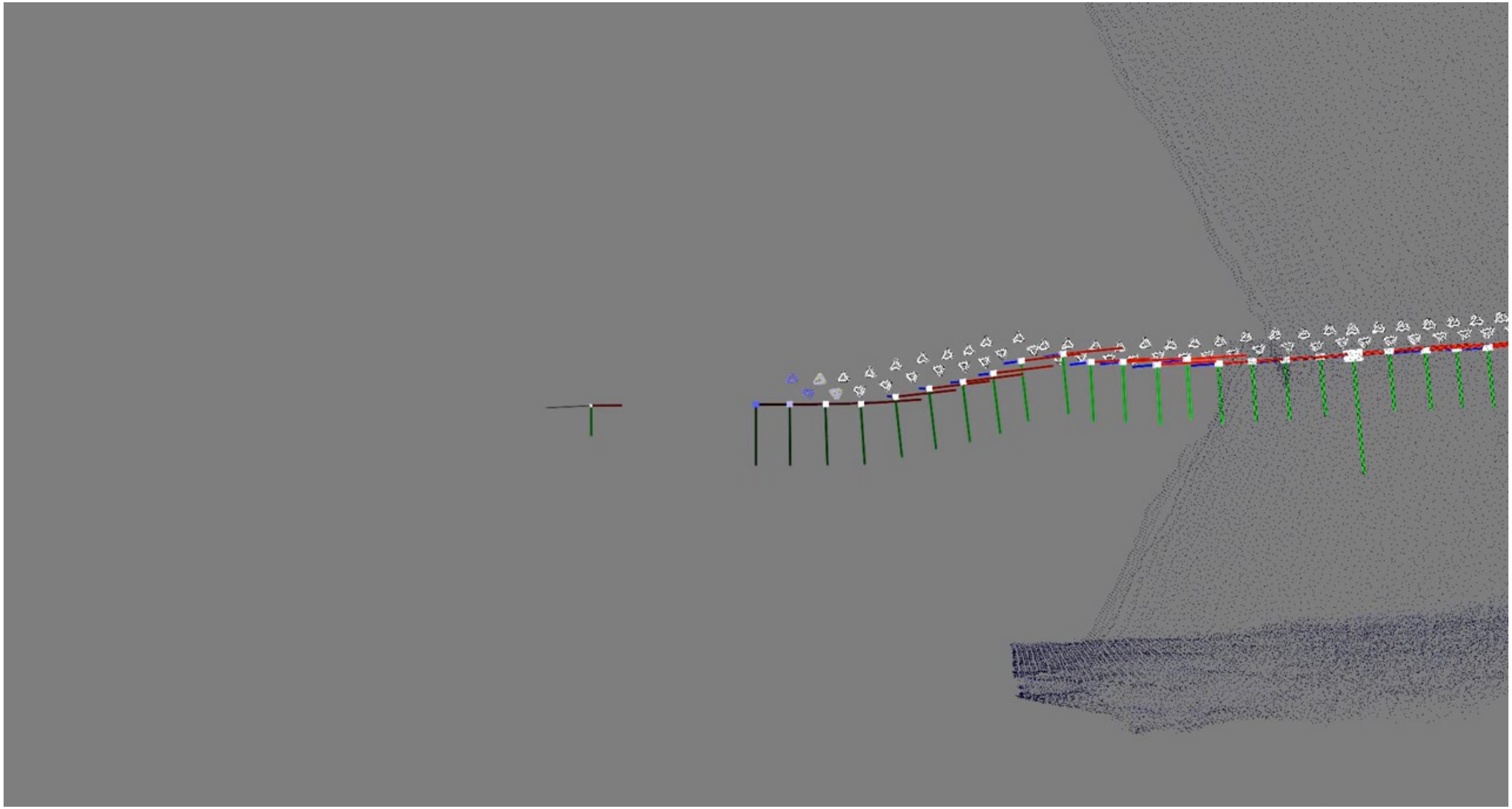
Use the Iterative Closest Point (**ICP**) algorithm to align scans, hence to compute **transformations** (i.e., relative motions) between robot positions.



Pose Graph with Scan Matching



Pose Graph with Scan Matching



SLAM Methods Recap

Kalman filter

- Online SLAM, parametric/Gaussian densities
- Fast but only for landmark-based maps, not robust, doesn't scale well

Particle filter

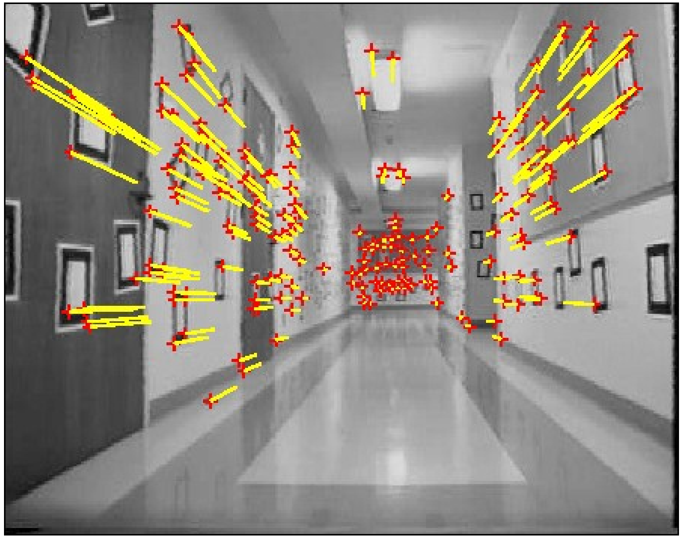
- Full SLAM, non-parametric, general densities
- Doesn't scale well, only for 2D motions

Graph-based

- Full SLAM, parametric/Gaussian densities
- Slow but very robust and well suited for large-scale problems

Hints on Visual SLAM

Visual SLAM is the process of incrementally estimating the pose of a camera and a map of the environment by **examining the changes that motion induces on the acquired images**.



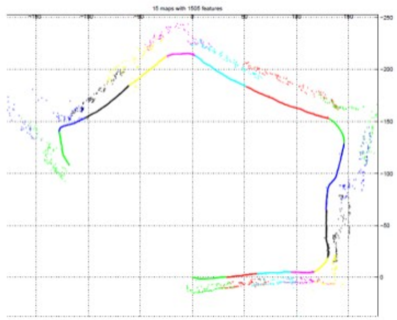
Visual Odometry VS Visual SLAM

Visual Odometry

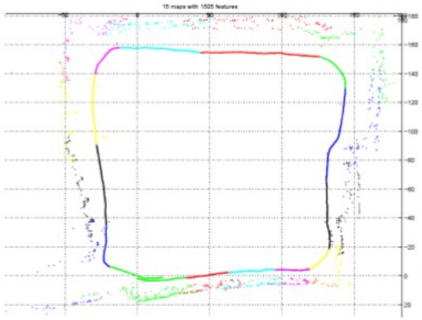
- Focus on incremental estimation/local consistency

Visual SLAM: Visual Simultaneous Localization And Mapping

- Focus on globally consistent estimation
- Visual SLAM = visual odometry + loop detection + graph optimization



Visual odometry

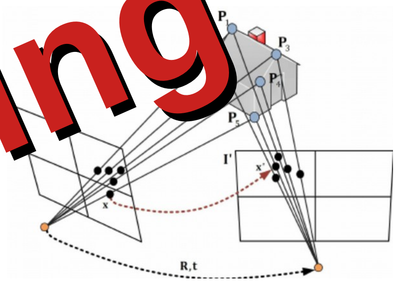


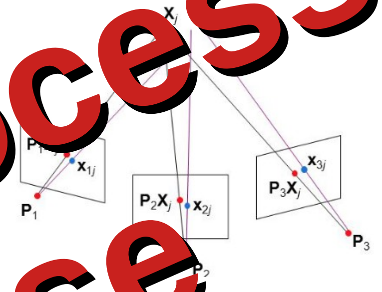
Visual SLAM

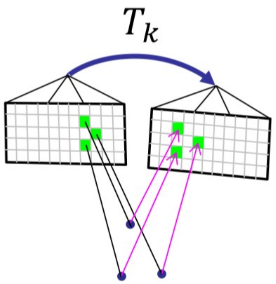
Image courtesy from [Clemente et al., RSS'07]

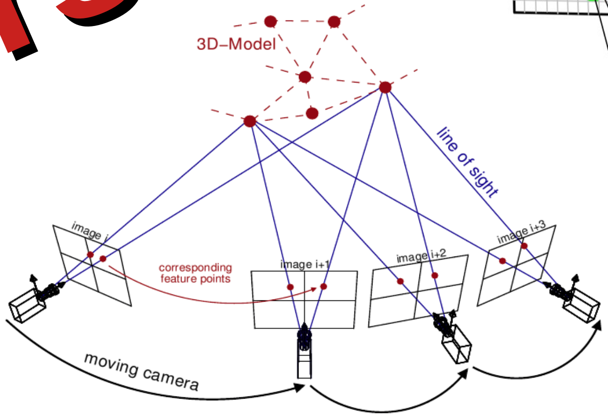
Visual Odometry/SLAM Approaches

- Epipolar geometry
- Filtering
- Direct approaches
- Bundle adjustment
- Hybrid approaches



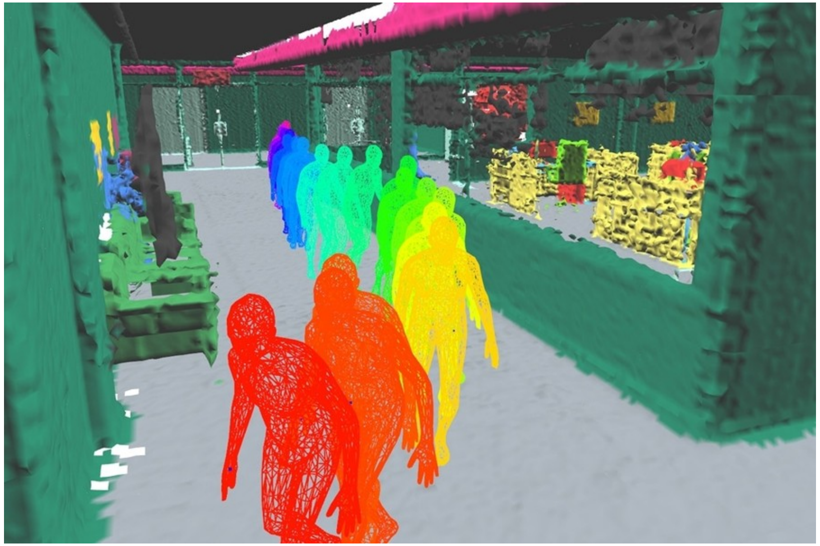




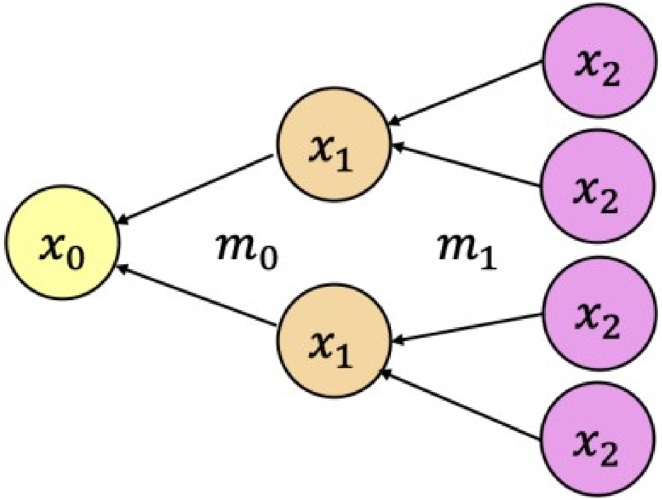


Current Trends in SLAM

SLAM in dynamic environments

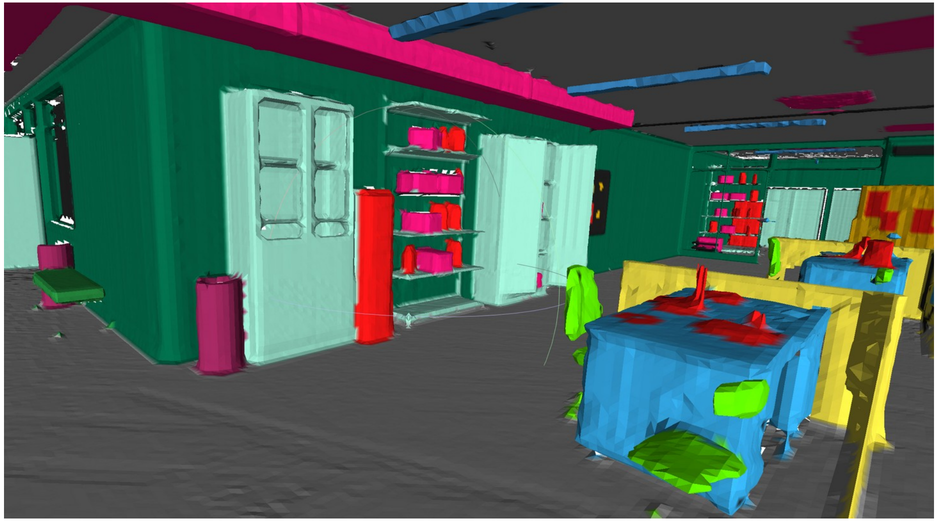


Multi-hypothesis SLAM

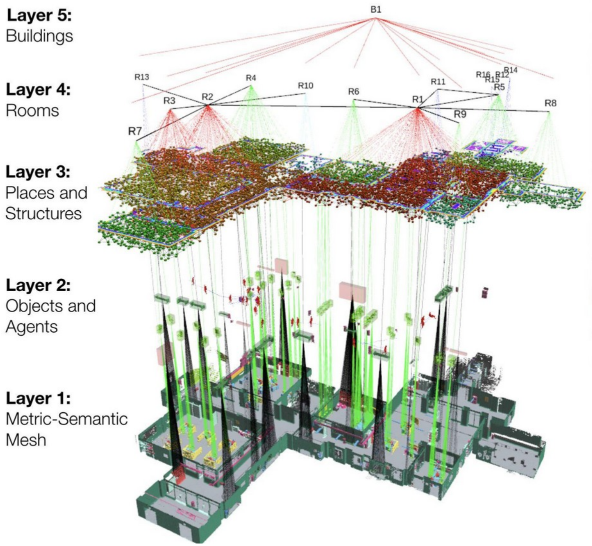


Current Trends in SLAM

Metric-Semantic SLAM

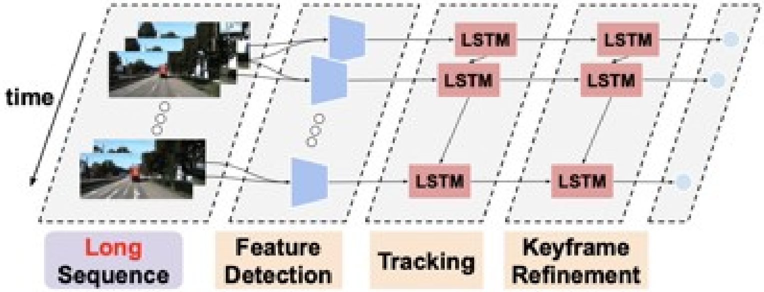


Hierarchical semantic SLAM

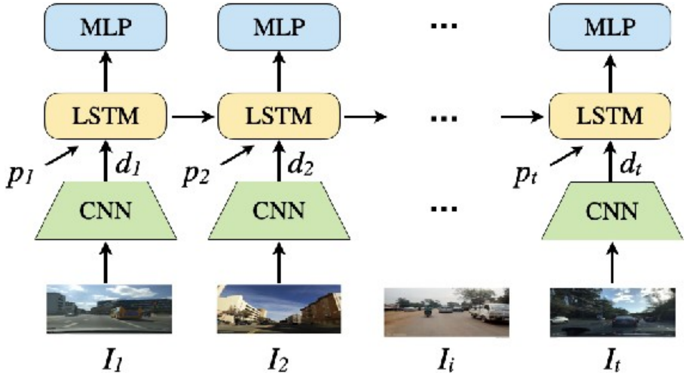


Current Trends in SLAM

Deep-learning based front-ends



Deep-learning based loop closure detection



Conclusions

- SLAM estimates the pose of a robot and the map of the environment at the same time.
- There are solid and established probabilistic solutions, in particular graph-based.
- There is still room for improvement, among other things, in terms of scalability and robustness against outliers.

Basic References

- Thrun et al. “Probabilistic Robotics”, MIT Press
- Springer “Handbook on Robotics”, Chapter on Simultaneous Localization and Mapping (1 st Ed: Chap. 37.1-37.2)
- Hugh Durrant-Whyte, Tim Bailey. “Simultaneous Localisation and Mapping (SLAM): Part I & II”, IEEE Robotics and Automation Magazine
- C. Cadena et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," in IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, Dec. 2016
- About Visual SLAM & Co.:
  - D. Scaramuzza, F. Fraundorfer: “Visual Odometry: Part I & II”, IEEE Robotics and Automation Magazine