AMR – 8. Simultaneous Localization and Mapping

Prof. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

Contents

Introduction to SLAM

Simultaneous Localization and Mapping – SLAM

- Is it possible to start at an unknown initial location, in an unknown environment and still incrementally build a map of the environment and at the same time use the map to determine the vehicle location?

- "Chicken and egg problem?"

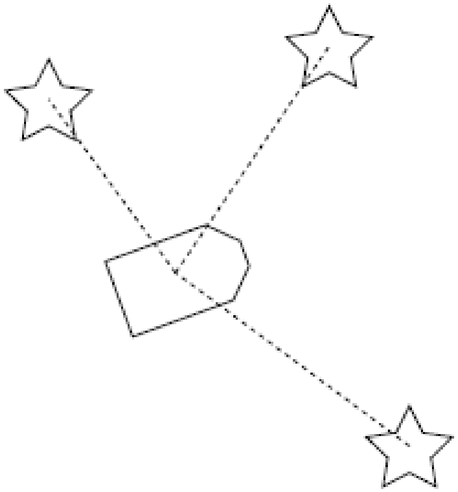
Simultaneous Localization and Mapping – SLAM

- The solution to the SLAM problem is, in many respects, a “Holy Grail” of the autonomous vehicle research community, as the ability to build a map and navigate simultaneously would indeed make a robot “autonomous”.
(Newman 1999, Leonard 2000, Thrun 2001)
- There is a large amount of potential applications
- It gives the vehicle real autonomy
- A solution is indeed possible

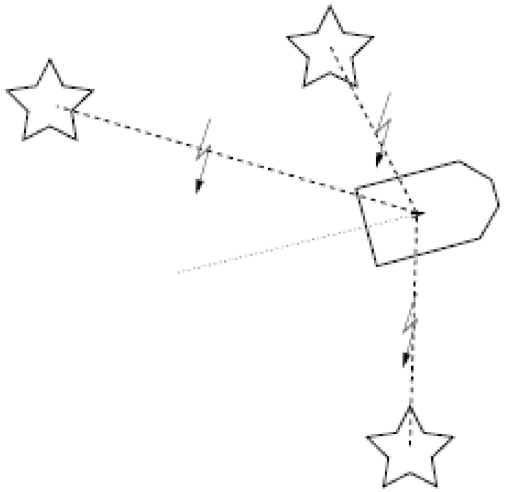
The General Approach

Tracking in a Local Feature Map

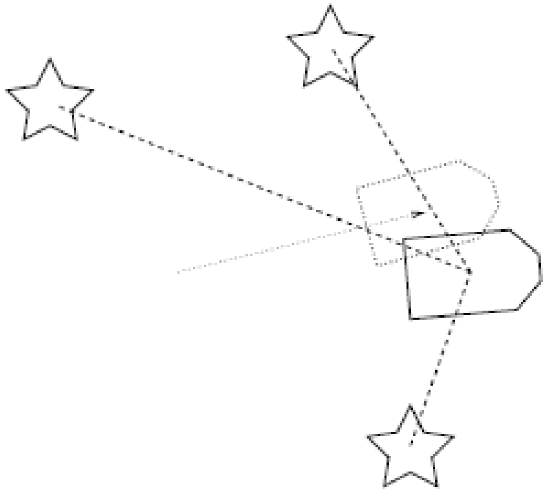
- Prerequisites: Extraction of features to identify landmarks
- Landmarks relative to the robot → local feature map
- Odometry predicts movements
- Local map allows correction of current pose



Local feature map



Odometry



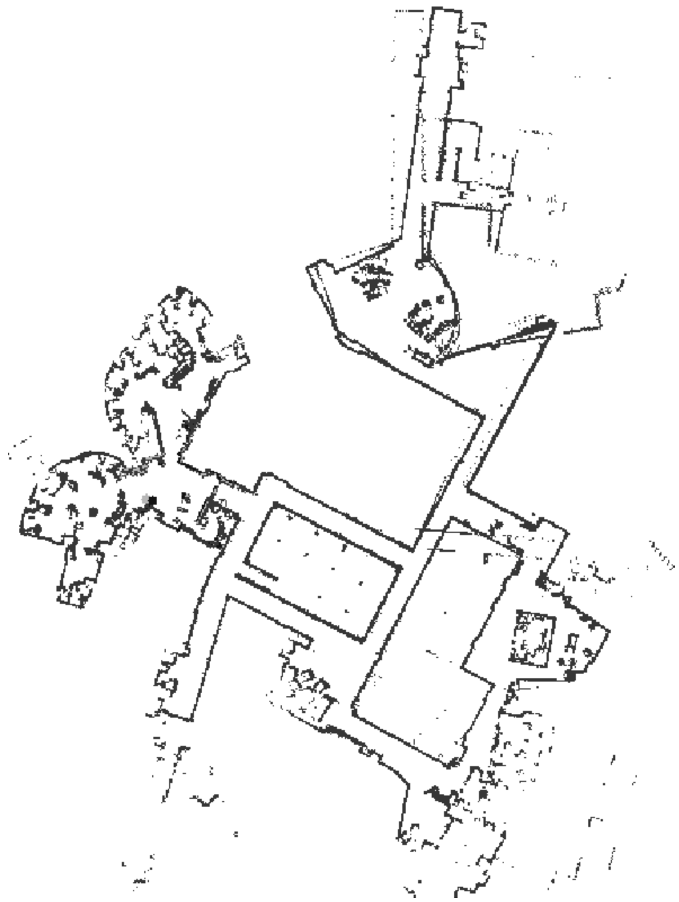
Correction using the map

Extending to a Global Map

- The initial local feature map contains the origin of the global map
- Adding new features to this map until the whole working space is covered creates the global map
- But …
  - Even after relocalization small errors persist due to the sensor systems used
  - Thus, adding new features based on relocalization results in errors that accumulate

Erroneous Maps from Simple Integration

Raw measurements

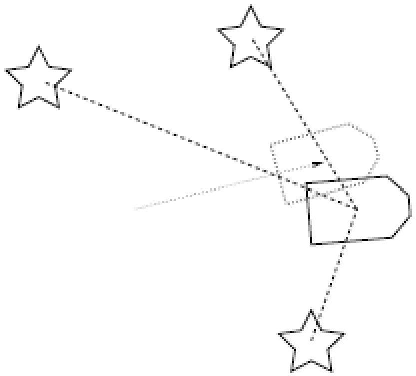


Corrected map

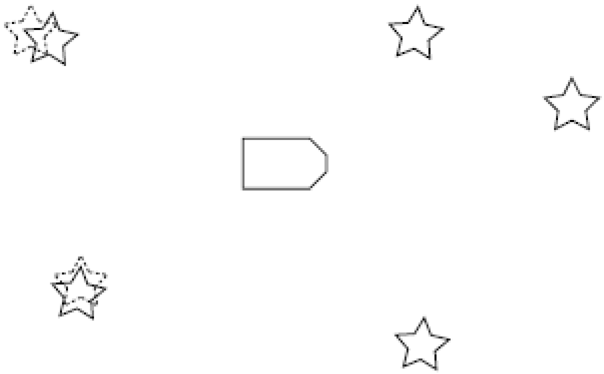
General Idea of the Solution

- Make small movements
- Stay within the range of as many known features as possible
- That allows a proper alignment of new features!

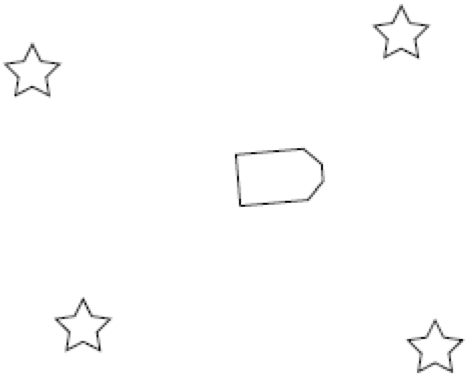
Proper Extension of the Map



Local map
from before



New local map with
displacement



Global map
after alignment

Open Question

How can we align
and merge local maps?

Registering of Point Clouds

The Benefit of Point Clouds

- The histogram method works with discrete distributed angles and therefore requires a structured environment (line features)
- If this is not available it is often possible to extract point features
- Using a distance sensor (like a laser scanner), the raw sensor data already is such a point cloud
- Finding the transformation that matches one point cloud with another is called Registration

Mean Squared Error

- Given two point sets, M and D with $|M| = |D|$ that correspond (i.e. every point m_i matches with one point d_i), a similarity measure describes how much these pairs differ
- A suitable similarity measure is the MSE

$$MSE = \frac{1}{|M|}\sum_{i=1}^{|M|} \|m_i - d_i\|^2$$

- Registration means to minimize the MSE!

The Iterative Closest Point Algorithm (ICP)

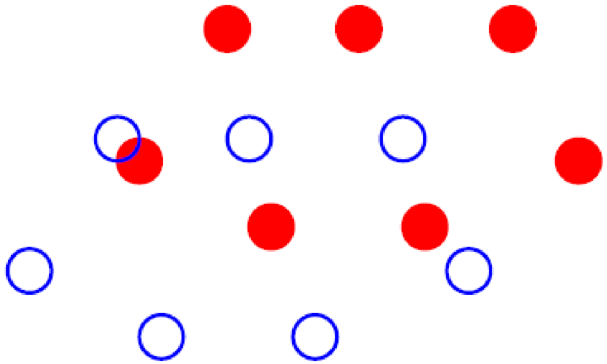
The ICP algorithm computes the transformation that minimizes the MSE as follows:

1) Choose as corresponding pairs those with the minimal Euclidean distance

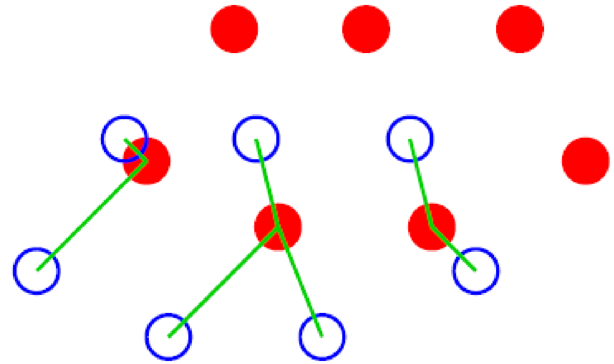
2) Calculate a rotation and transformation that minimizes the MSE of these pairs

3) Apply this transformation and repeat until the MSE falls below a limit

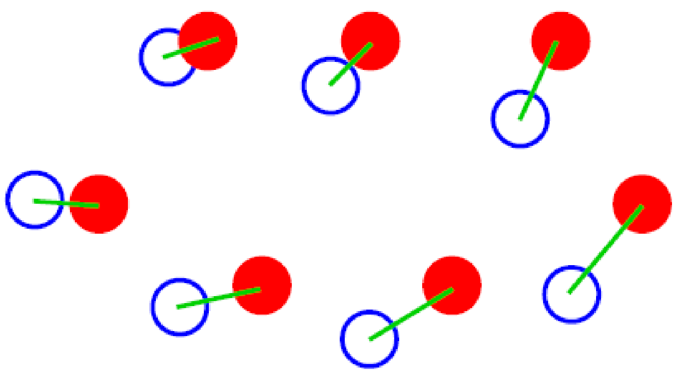
ICP Example



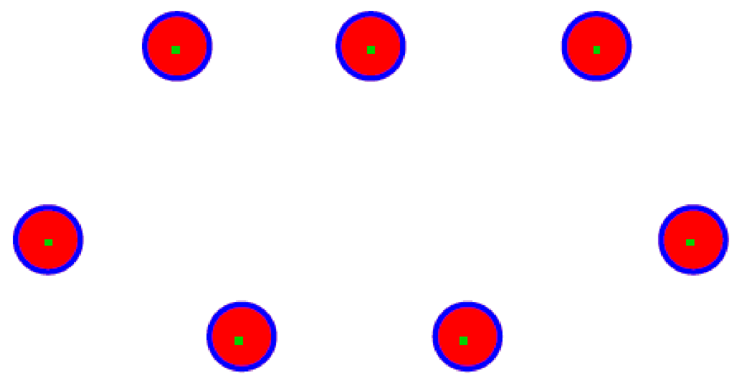
Initial state



First guess



Minimization and better guess



Solution

The Correspondence Assumption

- The correspondence of m_i and d_i is expressed by

$$m_i = Rd_i + T + \varepsilon_i$$

- Rotation matrix R
- Translation vector T
- Noise vector ε_i
  - Reflects measurement errors
  - The point clouds will not completely match

Application to the MSE

The optimal transformation $(\hat{R}, \hat{T})$ maps D to M while minimizing the MSE

$$MSE = \frac{1}{|M|} \sum_{i=1}^{|M|} \left\| m_i - \hat{R} d_i - \hat{T} \right\|^2$$

Centering the Point Clouds

In the end both point sets should have the same centroid:

- Both point clouds can be centered by subtracting their centroid

$$\bar{x} = \frac{1}{|X|} \sum_{i=1}^{|X|} x_i$$

$$\bar{x}_i = x_i - \bar{x}$$

- Now there exists a rotation that maps one cloud to the other

MSE Without Translation

After elimination of the translation the MSE can be rewritten:

$$MSE = \frac{1}{|M|}\sum_{i=1}^{|M|}\left\|\bar{m}_i - \hat{R}\bar{d}_i\right\|^2$$

$$= \frac{1}{|M|}\sum_{i=1}^{|M|}\left(\bar{m}_i^T\bar{m}_i + d_i^T\bar{d}_i - 2\underbrace{\bar{m}_i^T\hat{R}\bar{d}_i}_{*}\right)$$

($*$ must be maximized)

Scalarproduct and Trace of Outer Product

- The scalarproduct of two vectors a and b is

$$a^T b = \sum_i a_i b_i$$

- The trace of their outer product is

$$\mathrm{tr}(ba^T) = \mathrm{tr}\begin{pmatrix} b_1 a_1 & \cdots & b_1 a_i \\ \vdots & \ddots & \vdots \\ b_i a_1 & \cdots & b_i a_i \end{pmatrix} = \sum_i b_i a_i$$

Revised Maximization Problem

Maximize:

$$\sum_{i=1}^{|M|} \bar{m}_i^T \hat{R} \bar{d}_i = \mathrm{tr}\left(\sum_{i=1}^{|M|} \hat{R} \bar{d}_i \bar{m}_i^T \right)$$

$$= \mathrm{tr}\left(\hat{R} H\right)$$

$$H = \sum_{i=1}^{|M|} \bar{d}_i \bar{m}_i^T$$

Singular Value Decomposition

- Let A be a real $(m \times n)$-matrix of rank r
- A decomposition in the form of

$$A = USV^T$$

with orthogonal squared matrices U and V and a real diagonal $(m \times n)$-matrix

$$S = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ & 0 & & 0 \end{pmatrix}$$

is called Singular Value Decomposition of A

- $\sigma_1 \geq \cdots \geq \sigma_r$ are positive

SVD Applied

- The SVD of H yields

$$H = USV^T$$

- And with $X = VU^T$ we get

$$\begin{aligned} XH &= XUSV^T \\ &= VU^T USV^T \\ &= VSV^T \end{aligned}$$

- XH is symmetric and positive definite
- $XH = AA^T$ exists (Cholesky Decomposition)

Traces of Rotated Positive Definite Matrices

- Scalarproduct of rotated vectors (for every orthonormal matrix B)

$$a^T a \geq a^T B a$$

- Now, a_i being the i-th column of A

$$\mathrm{tr}(AA^T) = \sum_i a_i^T a_i \geq \sum_i a_i^T B a_i = \mathrm{tr}(BAA^T)$$

Finding the Optimal Rotation

- Applying the last slides gives us for every orthonormal matrix B

$$\mathrm{tr}(XH) \geq \mathrm{tr}(BXH)$$

- X itself is orthonormal and can be used as a rotation

$$\hat{R} = X = VU^T$$

Coping with Planar or Noisy Data

- For planar or noisy data SVD may compute a reflexion instead of a rotation
- This can be taken care of:

$$\hat{R} = V \begin{pmatrix} 1 & & & \\ & \ddots & & 0 \\ & & 1 & \\ & 0 & & s \end{pmatrix} U^T$$

$$s = \begin{cases} 1, & \det(H) \geq 0 \\ -1 & \text{else} \end{cases}$$

Computing the Translation

The missing translation $\hat{T}$ can easily be computed after applying $\hat{R}$

$$\hat{T} = \bar{m} - \hat{R}\bar{d}$$

The Outlier Problem

- ICP tries to match each point in D with one point in M
- So outliers (points that do not correspond) must be detected and filtered before applying the algorithm

Example: The Projection Filter

- The scan S of a $2\mathrm{D}$ laser range finder can be defined in the form of polar coordinates

$$S = \{s_i = (r_i, \alpha_i) \mid 0 \le i < n\}$$

$$n = \frac{360\,°}{\Delta\alpha}$$

- The single measurements are ordered by their angles

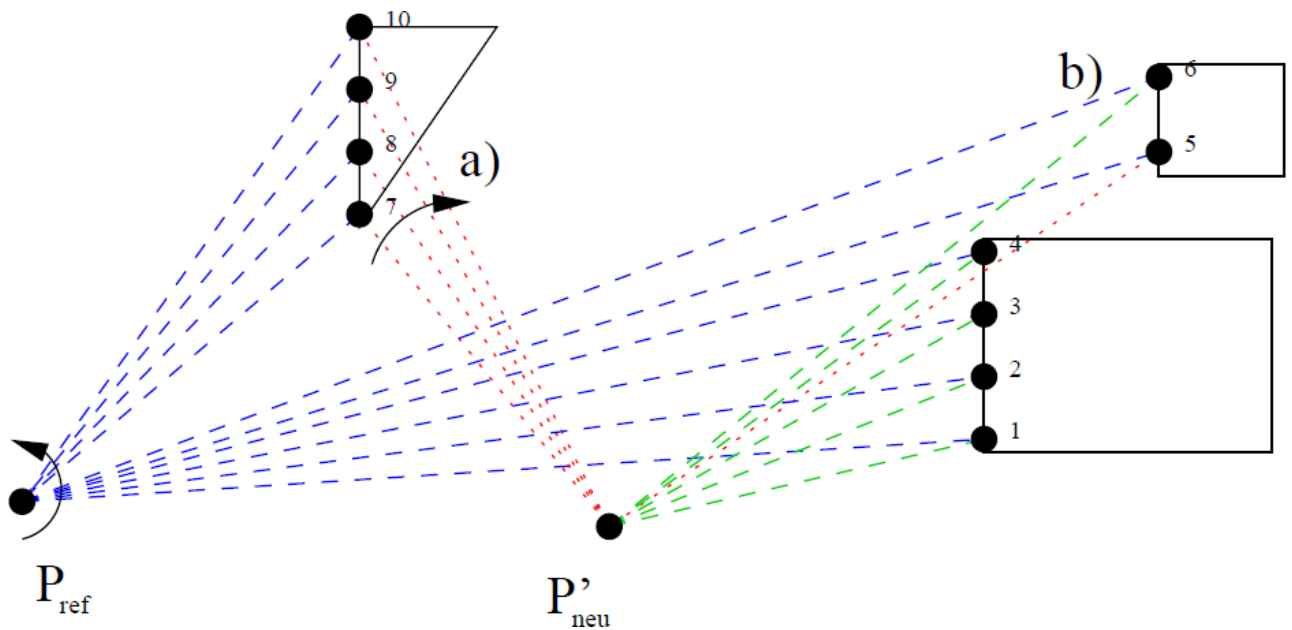
$$\alpha_i < \alpha_k \Leftrightarrow i < k$$

Example: The Projection Filter

- Let S_{ref} be a reference scan taken from the position P_{ref}
- Let S be a second scan from the estimated position P'_{new}
- The projection filter checks which scan points of S_{ref} are visible P'_{new}

Example: The Projection Filter

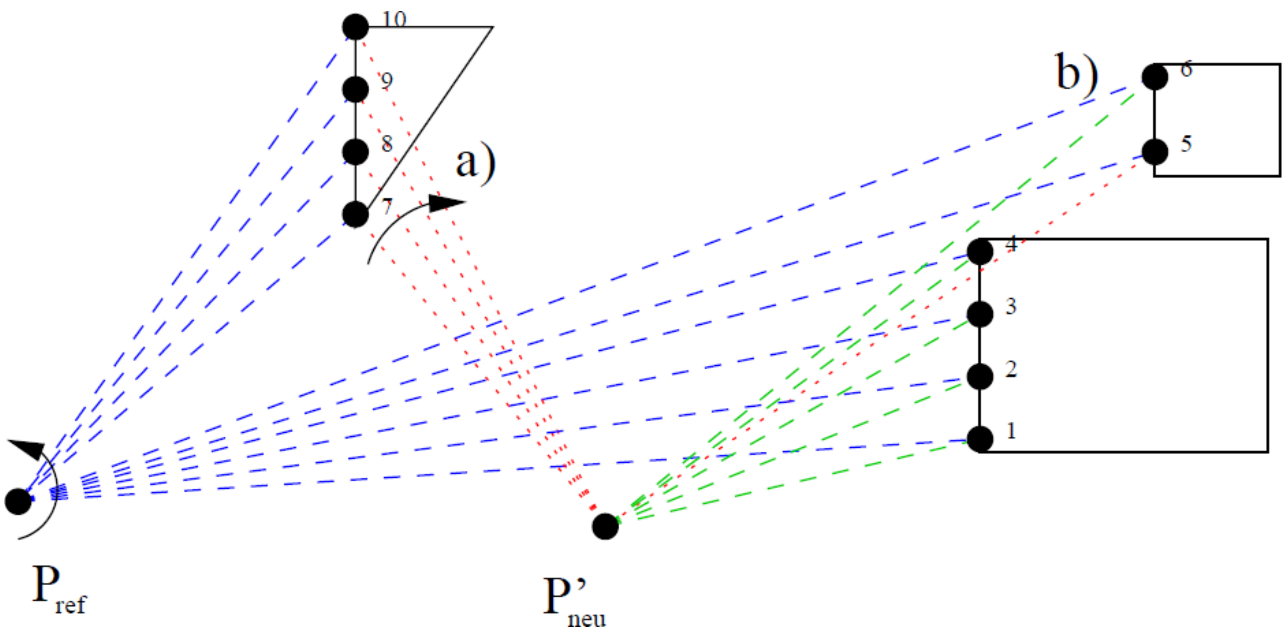
a) Use the order of scan points to determine points with reversed order. These are on the faces that pointed towards P_{ref} but point backwards for P'_{new}



The projection filter

Example: The Projection Filter

b) In case of multiple points on one straight line connecting a point of S with P'_{new}, only the first point (closest distance to P'_{new}) can be visible. Little deviations of this line must be taken into consideration

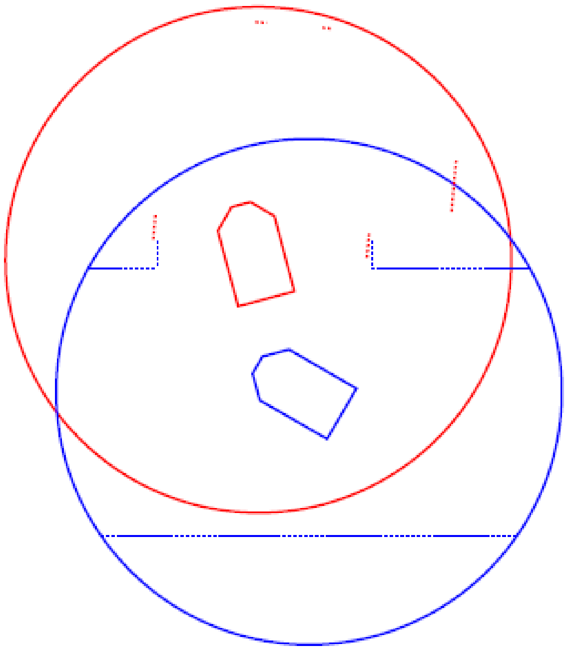


The projection filter

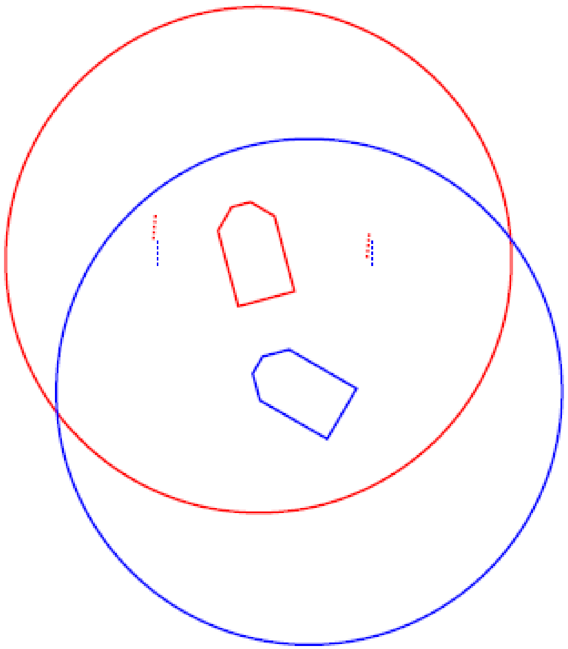
Example

Moving through room 48-358

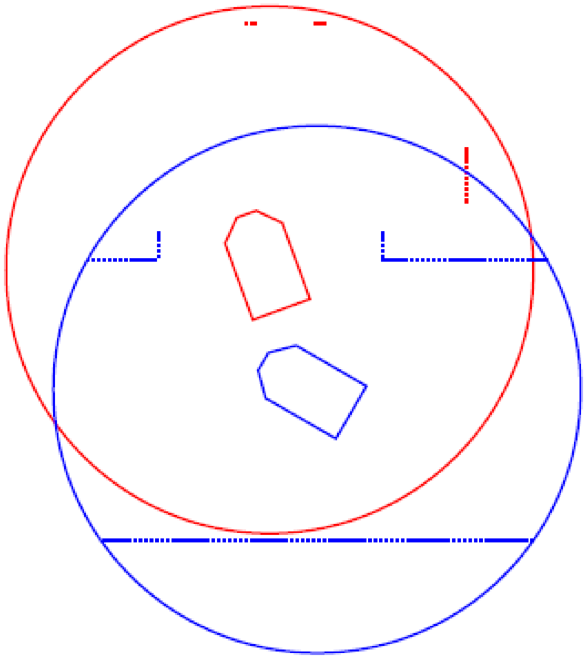
Example: A Robot Moving Through an Office



Robot moved from blue to red with scanning radius

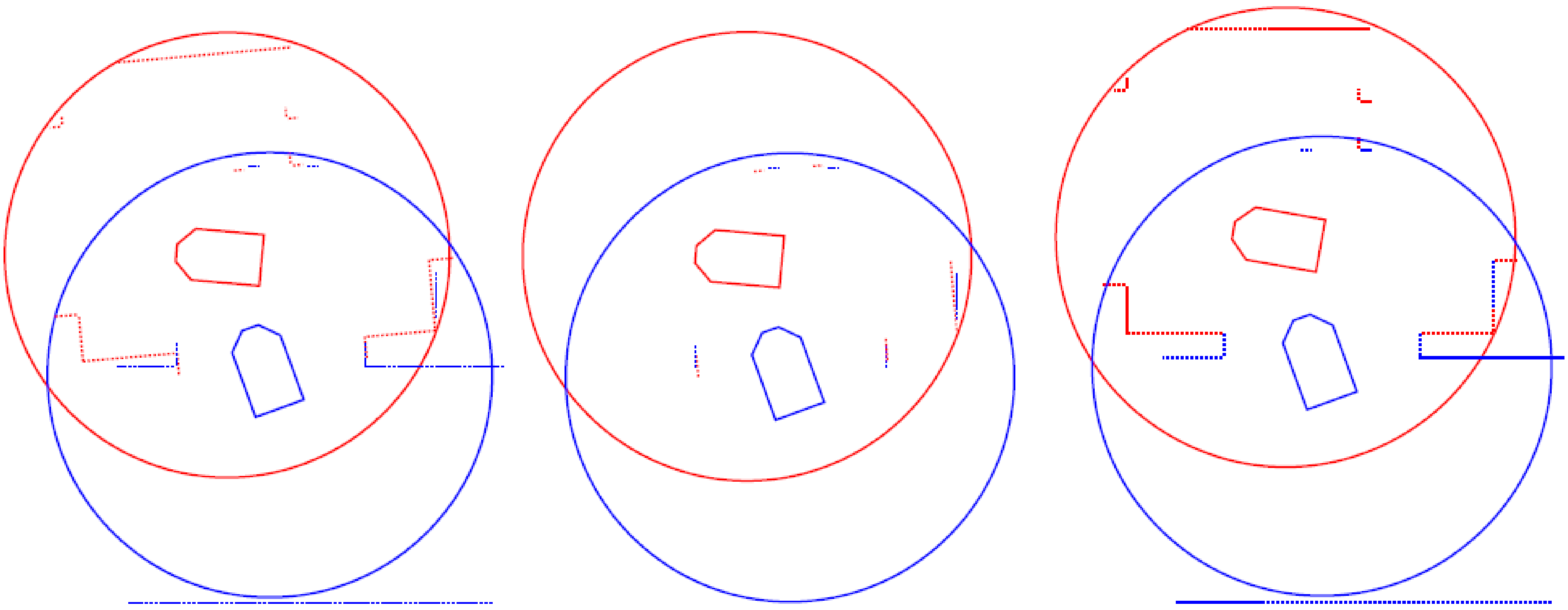


After applying the projection filter



After executing ICP

Example: A Robot Moving Through an Office

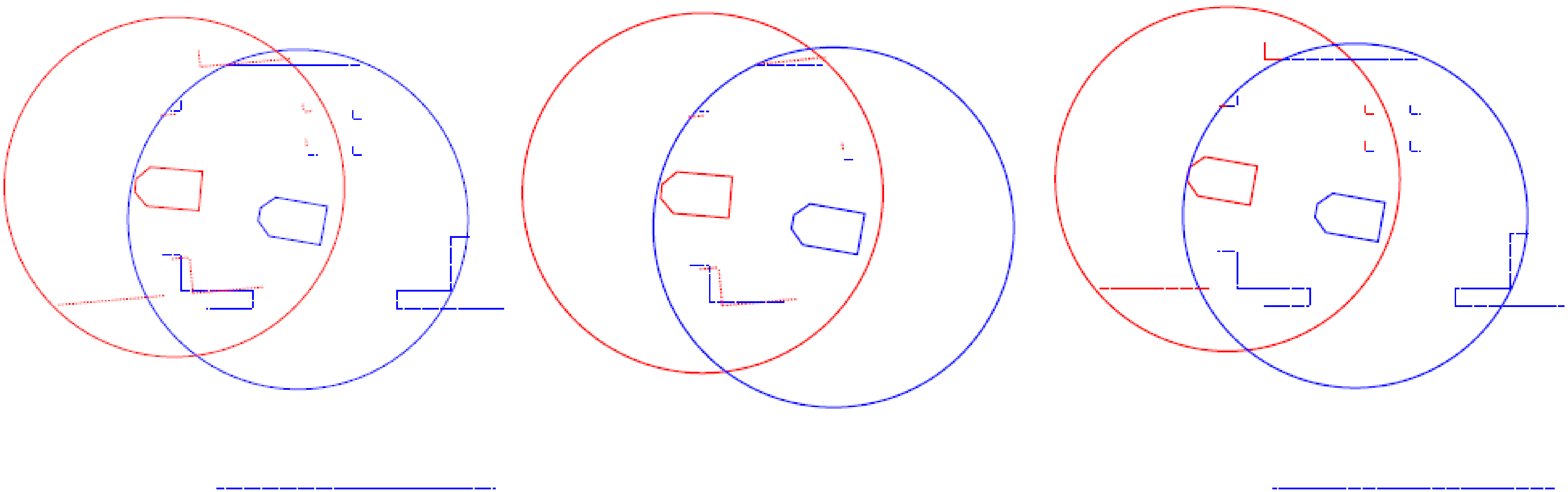


Robot moved from blue to red with scanning radius

After applying the projection filter

After executing ICP

Example: A Robot Moving Through an Office

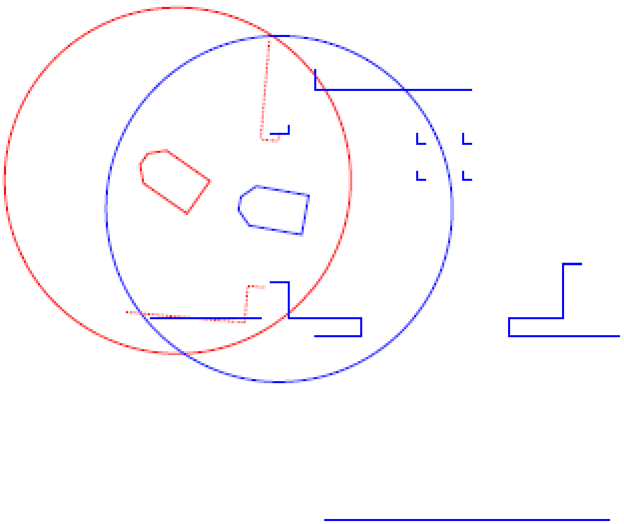


Robot moved from blue to red with scanning radius

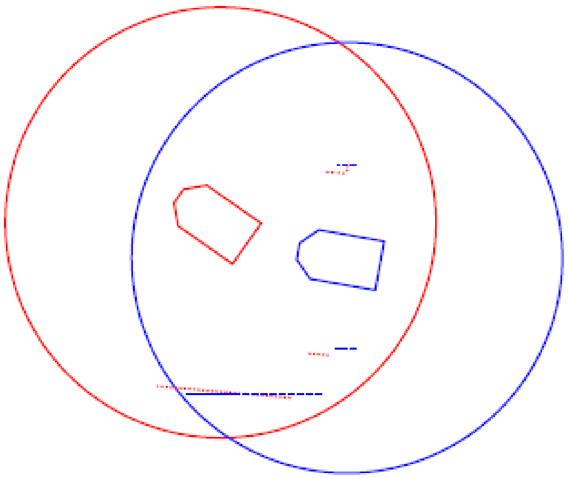
After applying the projection filter

After executing ICP

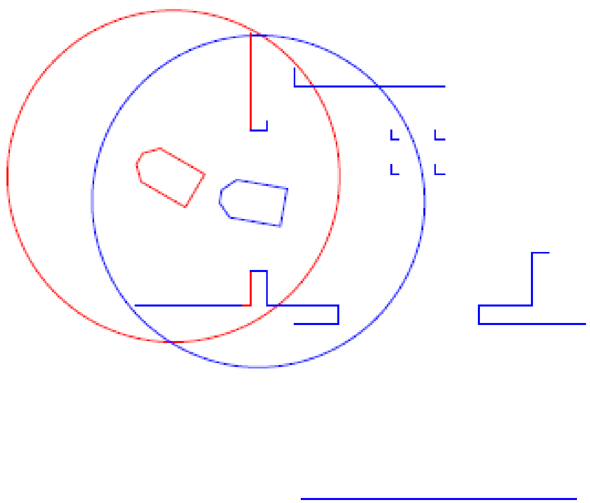
Example: A Robot Moving Through an Office



Robot moved from blue to red with scanning radius

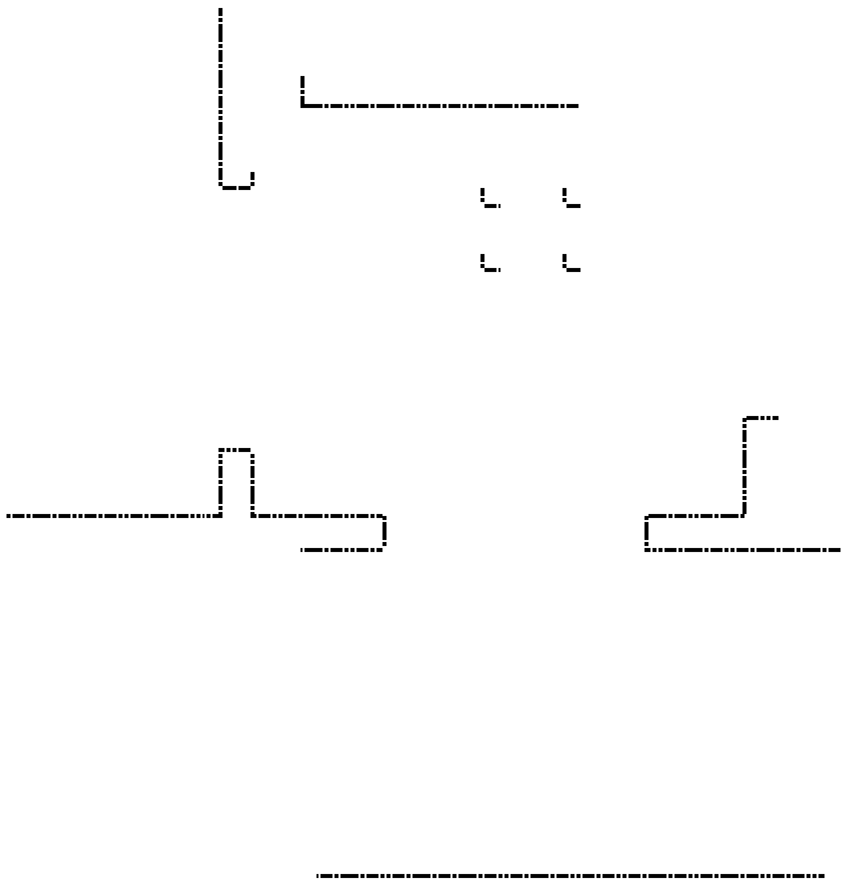


After applying the projection filter



After executing ICP

Example: Resulting Map



Loop Closing

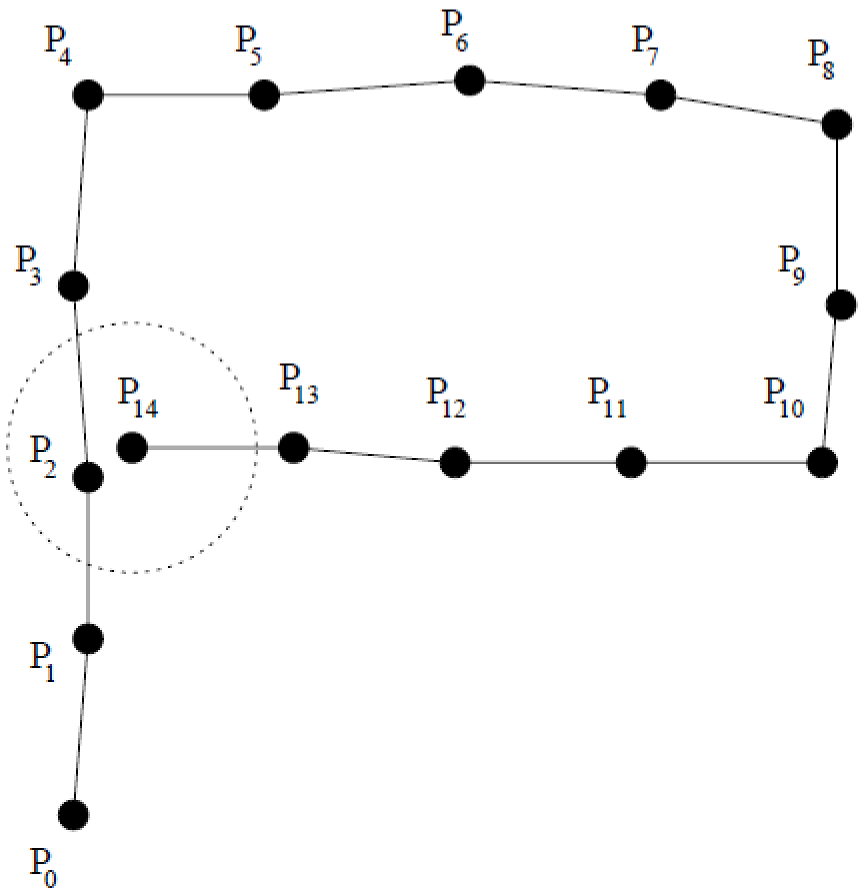
Loop Closing

- When implementing pixel-based algorithms do not integrate the results in one global pixel map
- Instead keep the local maps and just store them in a graph containing the transformations from one map to another
- If there are still errors they will become visible when driving in a loop and visiting a known place without recognizing that

Loop Closing – Solution

- To recognize a closed loop, a distance circle can be used
- The square of the pose displacement during map extension is chosen as radius
- If there is one other old node within this circle, a loop candidate exists
- Recognized loops can be used for error backpropagation and corrections in the just created map

Example: Loop Closing



Probabilistic Approach

Why Probabilistic Methods?

- Main problems
  - Unknown positioning errors
  - Noisy feature measurements
- But
  - Errors follow certain characteristics
  - → Uncertainty model

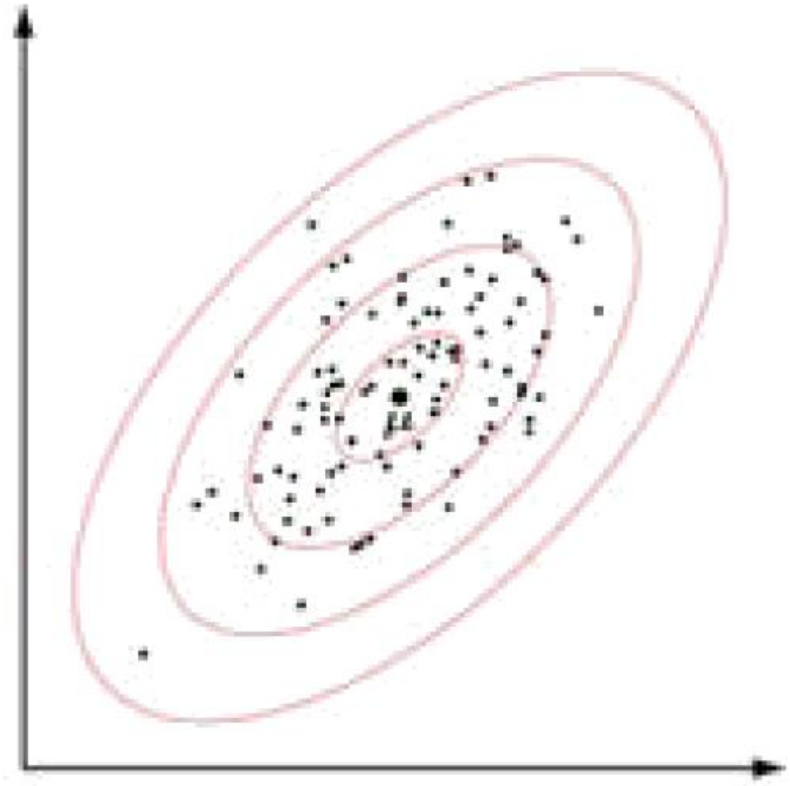
An Uncertainty Model

- Pose $d_i = \hat{d}_i + \tilde{d}_i$
  - Measured $\hat{d}_i$
  - Error $\tilde{d}_i$
- Expectation

$$\mu_i = E\left(\tilde{d}_i\right) = E\left(\left[d_i - \hat{d}_i\right]\right)$$

- Variance

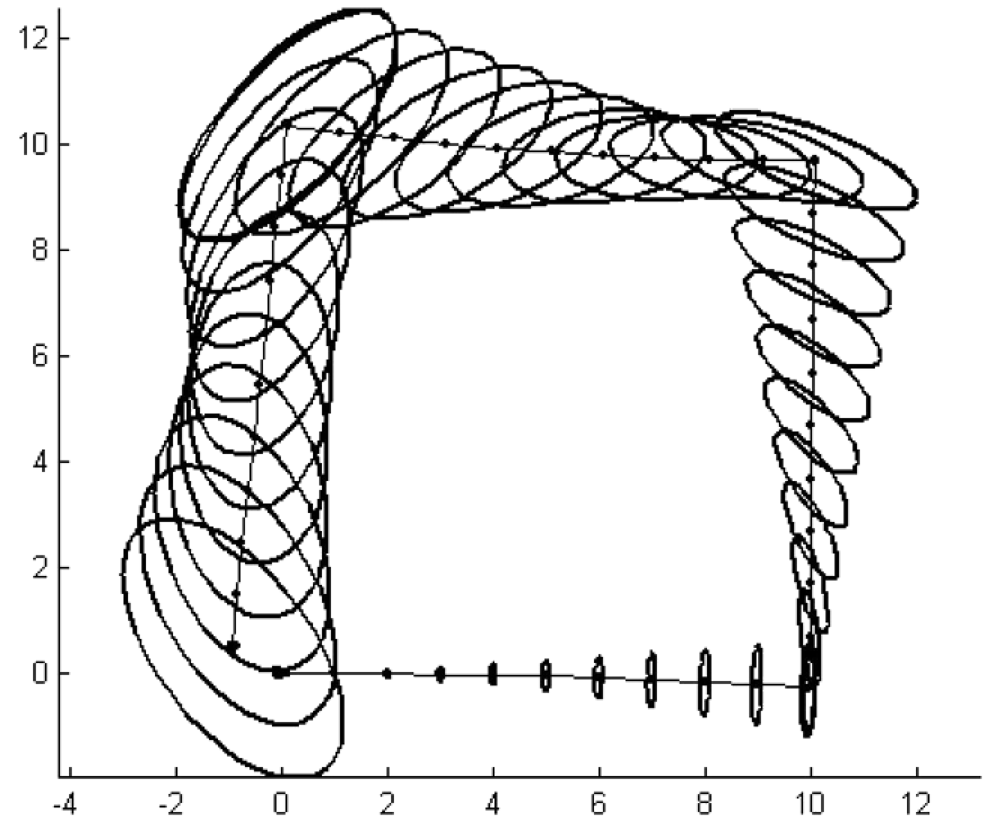
$$\sigma_i^2 = \mathrm{Var}\left(\tilde{d}_i\right)$$
$$= \mathrm{E}\left(\left[d_i - \hat{d}_i\right]\left[d_i - \hat{d}_i\right]^T\right)$$



Typical uncertainty distribution using odometry

Error Model for Differential Drive

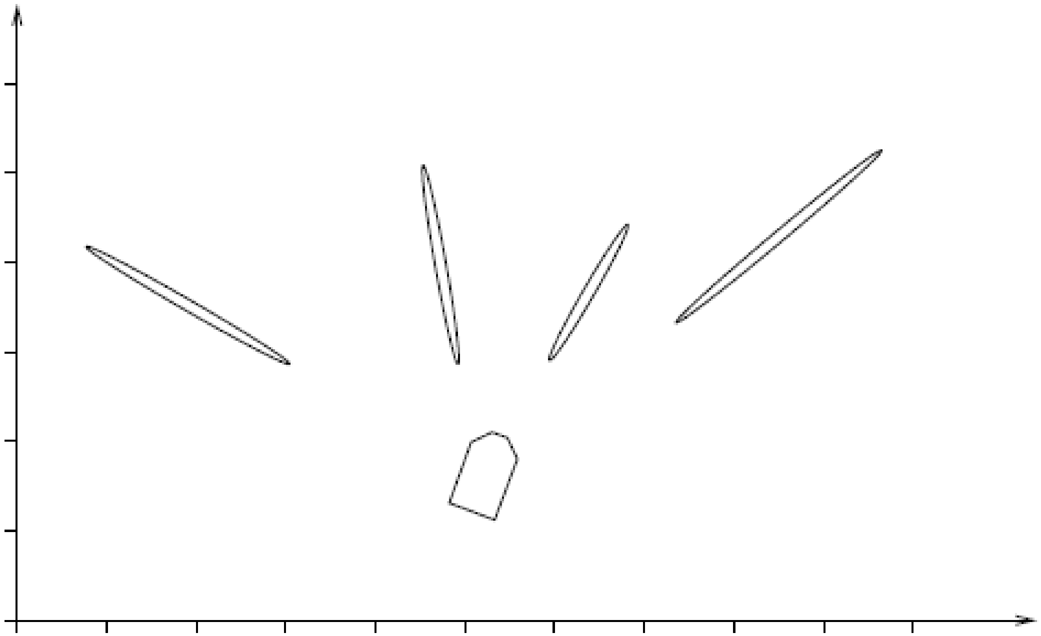
- Ellipsoid as model
- Low slipping
- Bigger rotational error



Typical error model
for differential drive

Error of the Sensor System

- Small angular error
- High distance error
- Elongate ellipsis



Sensor error model

Basic Idea of the Probabilistic Approach

- Use a coarse model of your situation to guess what could happen (Belief)
- Measure your environment and compare with your guess
- Improve your model over time
- From fuzzy guesses to a precise map
- Fuzziness → Robustness

Bayes‘ Theorem

- Let A and B be two random events and $p(A)$ and $p(B)$ be their a-priori probabilities
- A-posteriori probability of A after observing B

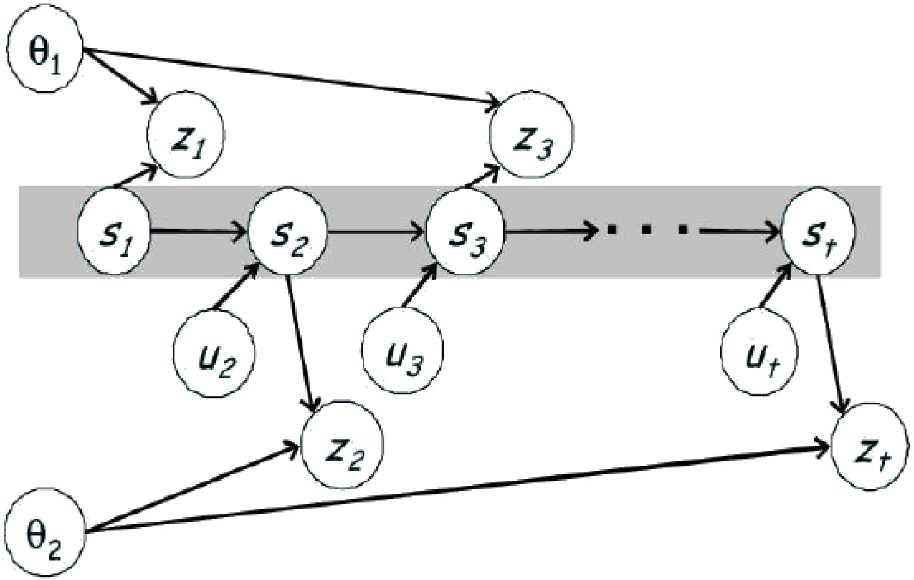
$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B)}$$

- The conditional Bayes’ Theorem says for additional background information E

$$p(A|B,E) = \frac{p(B|A,E)\,p(A|E)}{p(B|E)}$$

SLAM as Bayesian Network: FastSLAM

- Random variables
  - Position of the robot s_t
  - Control values u_t
  - Measured landmark positions z_t
  - Position of the landmarks θ_k
- The directed edges represent conditional dependencies



SLAM as Bayesian Network

Motion Model

- The robot's poses evolve according to the motion model

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

- s_t is a probabilistic function of …
  - control u_t
  - previous pose s_{t-1}

Measurement Model

- Landmarks are characterized by their location θ_k

$$p(z_t|s_t, \theta, n_t)$$

- θ is the set of all landmarks
- n_t is the index of the landmark observed as z_t at the time t
- The correspondence (value of n_t) is assumed to be known
- Serialize the observation of multiple landmarks at the same time!

Solving SLAM

- SLAM can be solved by calculation of

$$p(s^t, \theta | z^t, u^t, n^t)$$

- The superscript t describes a set of variables from time 1 to time t
- Individual landmark estimation problems are independent if path is known
- Solve $k + 1$ simpler problems

$$p(s^t, \theta | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_k p(\theta_k | s^t, z^t, u^t, n^t)$$

The Path Estimator

- A path estimator

$$p(s^t|z^t, u^t, n^t)$$

  is implemented using a particle filter
- Maintain a set S_t of particles representing the posterior distribution $p(s^t|z^t, u^t, n^t)$
- Each particle $s^{t,[m]}$ is a guess of the robot's path (superscript $[m]$ refers to the m-th particle)
- Each particle set S_t is calculated incrementally from the set S_{t-1}, a control u_t and a measurement z_t
- Generate a temporary guess $s_t^{[m]}$ using $p\left(s_t \middle| u_t, s_{t-1}^{[m]}\right)$ (dead reckoning)

Resampling

- Assume: S_{t-1} was distributed according to $p(s^{t-1}|z^{t-1},u^{t-1},n^{t-1})$
- S_t is distributed according to $p(s^t|z^{t-1},u^t,n^{t-1})$ as a proposal distribution
- This is achieved by sampling S_t from the temporary guesses with a probability that is proportional to an importance factor $w_t^{[m]}$

$$p\left(s^{t,[m]}\middle|z^t,u^t,n^t\right) = w_t^{[m]} p\left(s^{t,[m]}\middle|z^{t-1},u^t,n^{t-1}\right)$$

Resampling: Computation of the Weights

$$w_t^{[m]} = \frac{p\left(s^{t,[m]} \middle| z^t, u^t, n^t\right)}{p\left(s^{t,[m]} \middle| z^{t-1}, u^t, n^{t-1}\right)}$$

$$= \frac{p\left(s^{t,[m]} \middle| z_t, n_t, z^{t-1}, u^t, n^{t-1}\right)}{p\left(s^{t,[m]} \middle| z^{t-1}, u^t, n^{t-1}\right)}$$

$$\overset{Bayes}{=} \frac{\frac{p\left(z_t, n_t \middle| s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right)}{p(z_t, n_t | z^{t-1}, u^t, n^{t-1})} p\left(s^{t,[m]} \middle| z^{t-1}, u^t, n^{t-1}\right)}{p\left(s^{t,[m]} \middle| z^{t-1}, u^t, n^{t-1}\right)}$$

$$= \frac{p\left(z_t, n_t \middle| s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right)}{p(z_t, n_t | z^{t-1}, u^t, n^{t-1})}$$

$$\propto p\left(z_t, n_t \middle| s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right)$$

…

Resampling: Computation of the Weights

$$\ldots = p\left(z_t, n_t \middle| s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right)$$

$$\overset{Total\ prob.}{=} \int p\left(z_t, n_t \middle| \theta, s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right) p\left(\theta \middle| s^{t,[m]}, z^{t-1}, u^t, n^{t-1}\right) d\theta$$

$$\overset{Markov}{=} \int p\left(z_t, n_t \middle| \theta, s^{t,[m]}\right) p\left(\theta \middle| s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1}\right) d\theta$$

$$= \int p\left(z_t \middle| \theta, s^{t,[m]}, n_t\right) p\left(n_t \middle| \theta, s^{t,[m]}\right) p\left(\theta \middle| s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1}\right) d\theta$$

$$\propto \int p\left(z_t \middle| \theta, s^{t,[m]}, n_t\right) p\left(\theta \middle| s^{t-1,[m]}, z^{t-1}, u^{t-1}, n^{t-1}\right) d\theta$$

$$= \int p\left(z_t \middle| \theta_{n_t}^{[m]}, s^{t,[m]}, n_t\right) p\left(\theta_{n_t}^{[m]}\right) d\theta_{n_t}^{[m]}$$

The Landmark Estimators

The landmark estimators

$$p(\theta_k | s^t, z^t, u^t, n^t)$$

are implemented using Kalman filters

Particle	**Path**	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\cdots$	$\boldsymbol{\theta_k}$
1	s^t	μ_1, Σ_1	μ_2, Σ_2		μ_k, Σ_k
2	s^t	μ_1, Σ_1	μ_2, Σ_2		μ_k, Σ_k
$\vdots$					
m	s^t	μ_1, Σ_1	μ_2, Σ_2		μ_k, Σ_k

Add Landmarks

- If first added in t, the expectation and covariance of the landmark must be calculated
- With the average distance error σ_1 and average angular error σ_2 the observation noise is

$$R = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

- The observation can be described as

$$G = \begin{bmatrix} \cos\left(s_{t,3}^{[m]} + z_{t,2}\right) & -z_{t,1}\sin\left(s_{t,3}^{[m]} + z_{t,2}\right) \\ \sin\left(s_{t,3}^{[m]} + z_{t,2}\right) & z_{t,1}\cos\left(s_{t,3}^{[m]} + z_{t,2}\right) \end{bmatrix}$$

Add Landmarks

- Now the covariance is

$$\sum^{t,[m]} = GRG^T$$

- And the expectation

$$\mu_t^{[m]} = \begin{bmatrix} s_{t,1}^{[m]} + z_{t,1} \cos\left(s_{t,3}^{[m]} + z_{t,2}\right) \\ s_{t,2}^{[m]} + z_{t,1} \sin\left(s_{t,3}^{[m]} + z_{t,2}\right) \end{bmatrix}$$

Revisiting Landmarks

The observation can be predicted:

$$d^{[m]} = \begin{bmatrix} \mu_{t-1,1}^{[m]} - s_{t,1}^{[m]} \\ \mu_{t-1,2}^{[m]} - s_{t,2}^{[m]} \end{bmatrix}$$

$$z_t^{[m]'} = \left(\left| d^{[m]} \right|, \operatorname{atan2}\left(d_2^{[m]}, d_1^{[m]} \right) - s_{t,3}^{[m]} \right)$$

Revisiting Landmarks

The Jacobian Matrices w. r. t. to the vehicle and landmark states are:

$$H_{vehicle}^{[m]} = \begin{bmatrix} -\frac{d_1^{[m]}}{\left|d^{[m]}\right|} & -\frac{d_2^{[m]}}{\left|d^{[m]}\right|} & 0 \\ \frac{d_2^{[m]}}{\left|d^{[m]}\right|^{2}} & -\frac{d_1^{[m]}}{\left|d^{[m]}\right|^{2}} & -1 \end{bmatrix}$$

$$H_{landmarks}^{[m]} = \begin{bmatrix} \frac{d_1^{[m]}}{\left|d^{[m]}\right|} & \frac{d_2^{[m]}}{\left|d^{[m]}\right|} \\ -\frac{d_2^{[m]}}{\left|d^{[m]}\right|^{2}} & \frac{d_1^{[m]}}{\left|d^{[m]}\right|^{2}} \end{bmatrix}$$

Update the Path Estimation

- Now covariance can be predicted

$$\sum_{t}^{[m]'} = H_{landmarks}^{[m]} * \sum_{1}^{[m]} * H_{landmarks}^{[m]^T} + R$$

- Knowing the error with respect to the observation $\varepsilon^{[m]} = z_t - z_t^{[m]'}$ the weights $w_t^{[m]}$ are

$$w_t^{[m]} = \frac{e^{-\frac{\varepsilon^{[m]^T} \Sigma_2^{[m]'^{-1}} \varepsilon^{[m]}}{2}}}{2\pi\sqrt{\left|\Sigma_2^{[m]'}\right|}}$$

Update Landmarks

- Assume $n_t = k$ (Landmark θ_k is visible at time t)

$$
\begin{aligned}
p(\theta_k | s^t, z^t, u^t, n^t) &= p(\theta_k | z_t, s^t, z^{t-1}, u^t, n^t) \\
&= \frac{p(z_t | \theta_k, s^t, z^{t-1}, u^t, n^t)\, p(\theta_k | s^t, z^{t-1}, u^t, n^t)}{p(z_t | s^t, z^{t-1}, u^t, n^t)} \\
&\propto p(z_t | \theta_k, s^t, z^{t-1}, u^t, n^t)\, p(\theta_k | s^t, z^{t-1}, u^t, n^t) \\
&\overset{Markov}{=} p(z_t | \theta_k, s_t, u_t, n_t)\, p(\theta_k | s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1})
\end{aligned}
$$

- For $n_t \neq k$: θ_k not visible at time $t \rightarrow$ no change

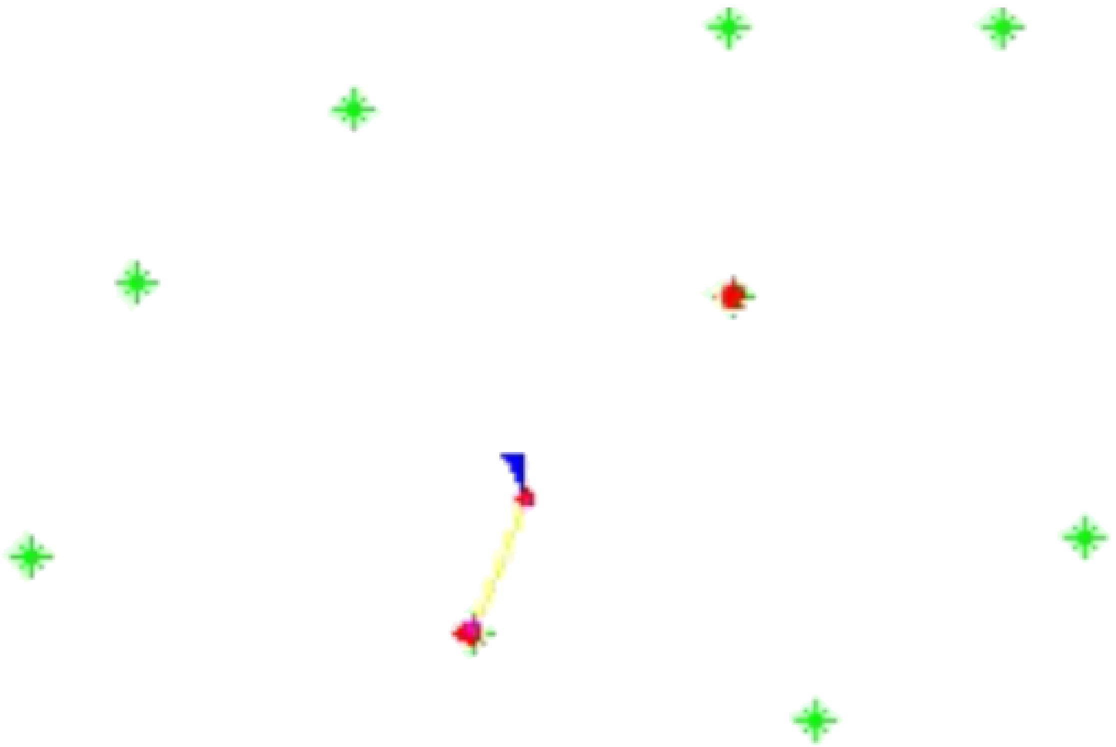
$$
p(\theta_k | s^t, z^t, u^t, n^t) = p(\theta_k | s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1})
$$

Update Landmarks

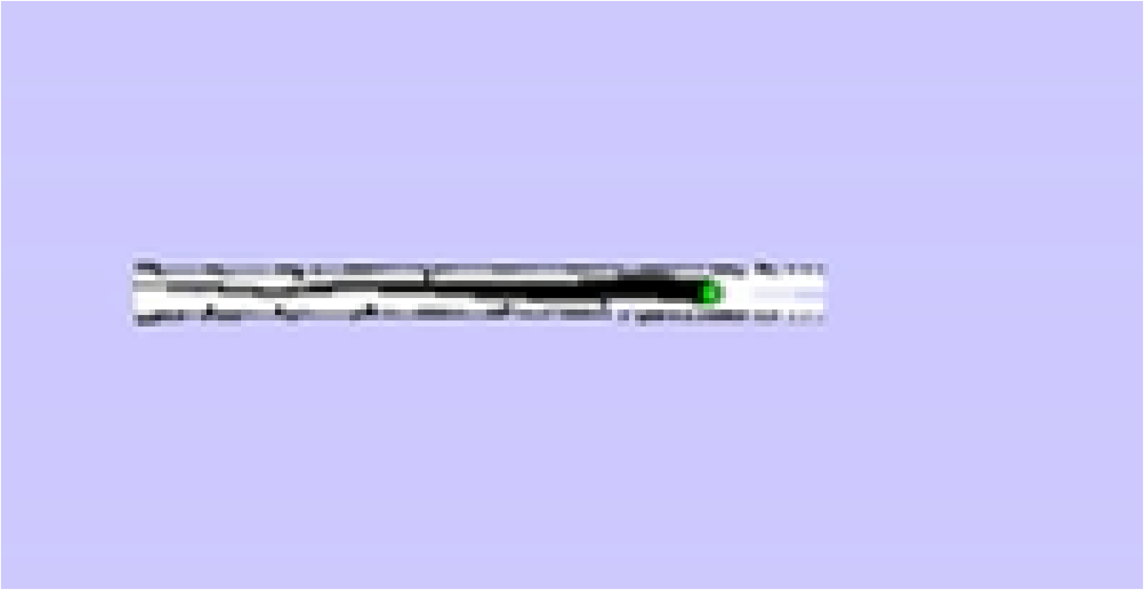
The landmark estimations can be updated using a Kalman Filter with:

- The prior state $\mu_1^{[m]}, \Sigma_1^{[m]}$
- The innovation $\varepsilon^{[m]}, R$
- The linearized observation model $H_{landmarks}^{[m]}$

FastSLAM Example



FastSLAM Example



Coming Next

Navigation