DD2426 – Robotics and Autonomous Systems
Lecture 6-7: Localization and Mapping

Patric Jensfelt

Kungliga Tekniska Högskolan
patric@kth.se

April 15 & 17,2007

Course admin

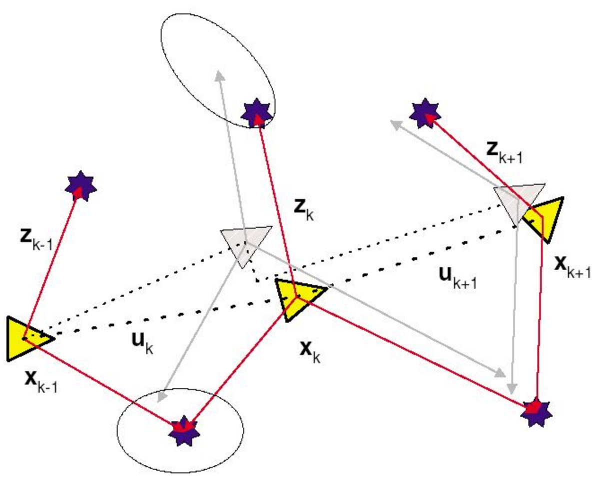
- Everyone completed milestone0 on time. Good work!
- but do not get overconfident, it gets more difficult

Localization and Mapping

- Fundamental competences in a mobile robot system
- Many task require knowledge of the position
- A map is often needed to plan tasks and stay localized

The Localization Problem

- Given: Map of the environment
- Want: Pose of the robot
- Problem: Estimate $p(x|\mathcal{M}, \{z_0, z_1, \ldots\})$



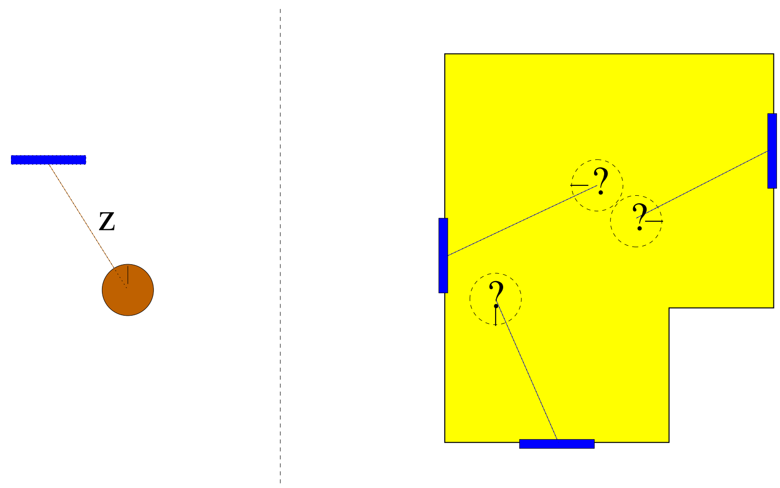
Picture from Durrant-Whyte[1]

Two sides of Localization

- Can identify two parts in the localization problem
- Depends on the prior knowledge
  - Global localization
  - Pose tracking

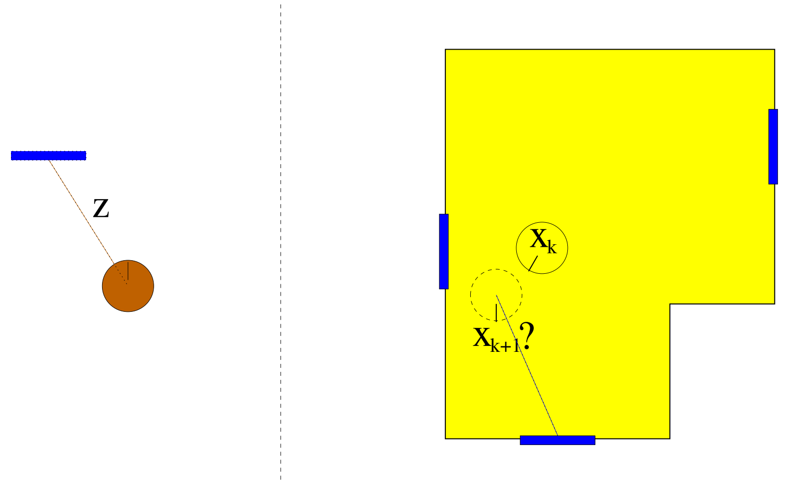
Global localization

- Find the pose of the robot without any prior knowledge
- Data can typically be explained in several ways
  $\Rightarrow$ ambiguities
- Can typically not be done without integrating information over time



Pose Tracking

- Robot knows where it was and just has to keep localized
- Simpler problem

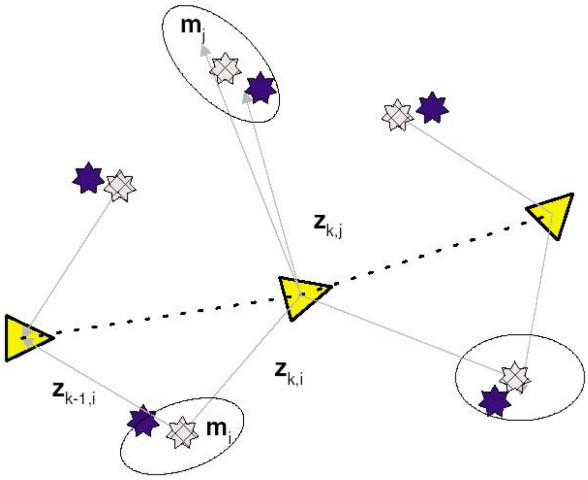


Kidnapping Problem

- Robot is abducted in tracking phase and put down somewhere else
- Hard problem
- Must first realize that it is lost
- Then perform global localization
- Example: Cleaning robot that is carried away to another room

The Mapping Problem

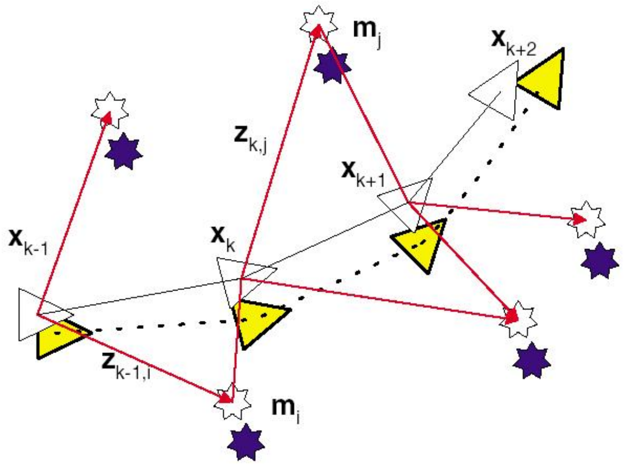
- Given: Pose of the robot
- Want: Map of the environment
- Problem: Estimate $p(\mathcal{M}|\{x_0, x_1, \ldots\}, \{z_0, z_1, \ldots\})$



Mapping is not a new problem!

Building a Map of an Unknown Enviroment

- Want: Pose of the robot
- Want: Map of the environment
- Problem: Estimate $p(\mathcal{M}, \{x_0, x_1, \ldots\}|\{z_0, z_1, \ldots\})$



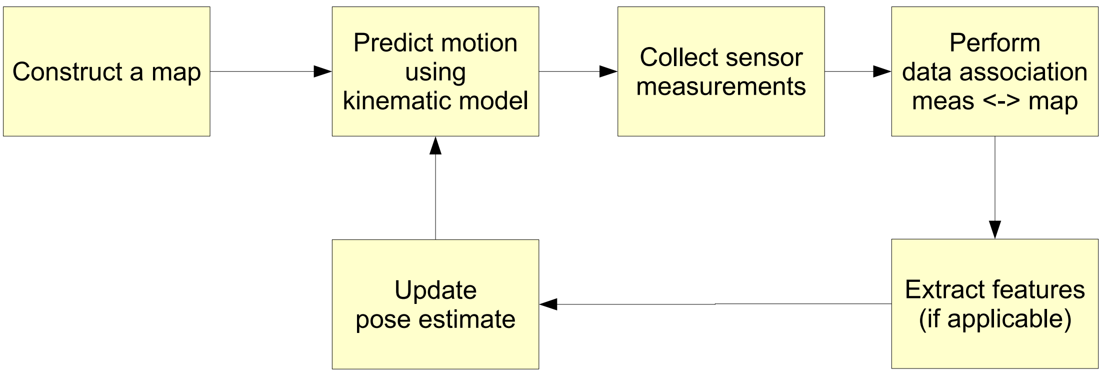
What is SLAM?

- Chicken and Egg problem
  - Need map to find pose!
  - Need pose to build map!
- Need to do them simultaneusly
  $\Rightarrow$ SLAM = Simultaneus Localization and Mapping

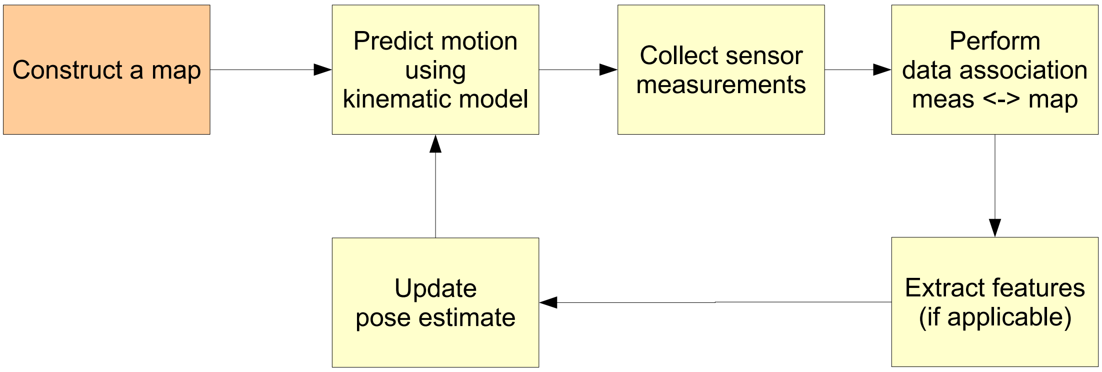
Design decisions

- How do we represent the environment?
- What sensors to use?
- How do we represent our knowledge about robot pose?
- What algorithms to use for updating the information?

Localization process cycle



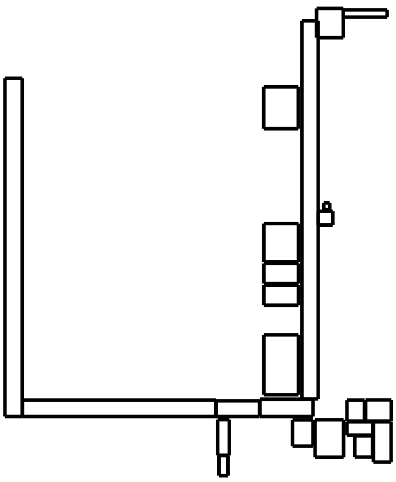
Map representation



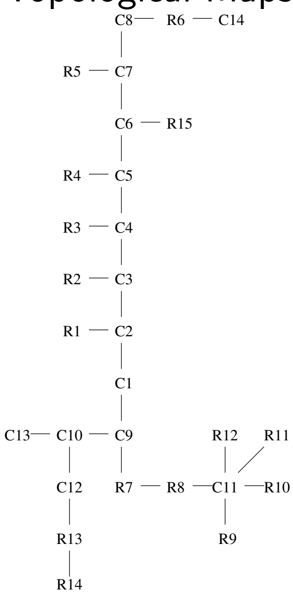
Map Representations

- Two Main Directions

Metric Maps



Topological Maps



Topological Map Representation

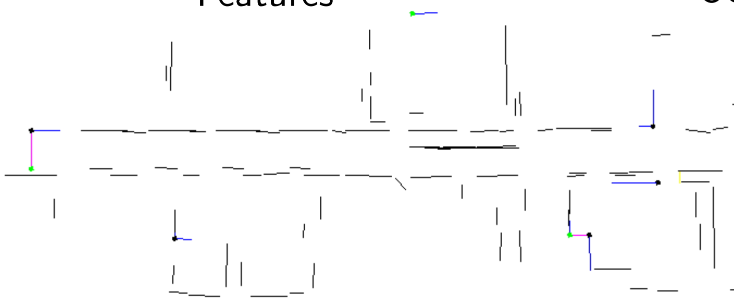
- Often defined by the structure of the environment
- Represented by a graph
- Each node corresponds to a place
- Connections between places corresponds to edges in the graph, could be doors, stairways, etc
- Requires that the robot can detect places
  Very hard problem in general
- Sometimes augmented with rough metric information

Topological Maps cont'd

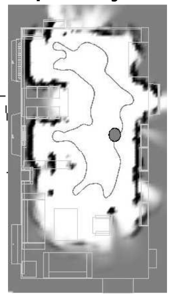
- Pros:
  - Scales very well with the size of the environment
  - Can be constructed without knowing the exact geometry
- Cons:
  - Coarse position information
  - Relies on place recognition which is HARD!

Different Metric Representations

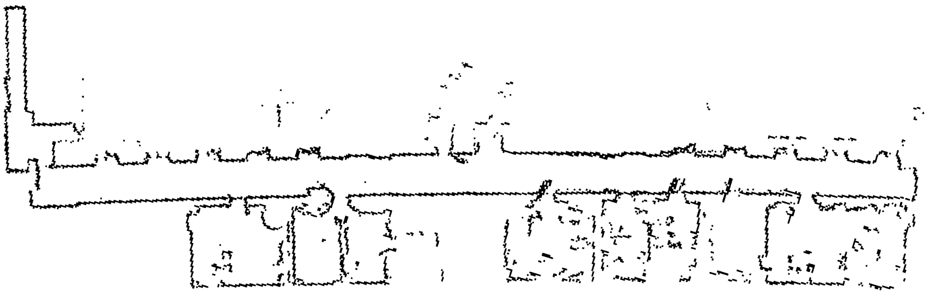
Features



Occupancy Grids



Appearance Based (raw data)



Feature Maps

- Map represented by a set of features

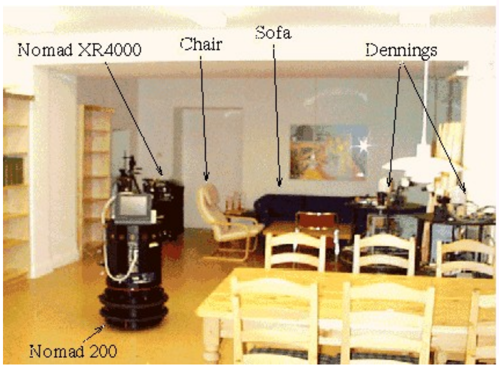
$$\mathcal{M} = \{m_j \mid j = 1, \ldots, M\}$$

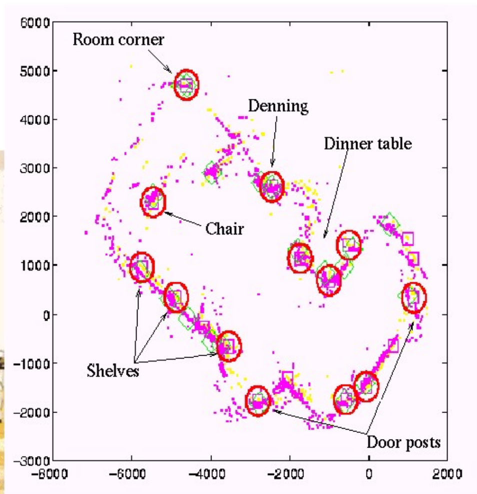
- Natural way to describe the world
- Has been used for a long time
- Examples of features:
  - Point, lines, plane
  - Edges, corners
  - Visual features

Example: Feature map

- Line map

Sonar Point Landmark Map





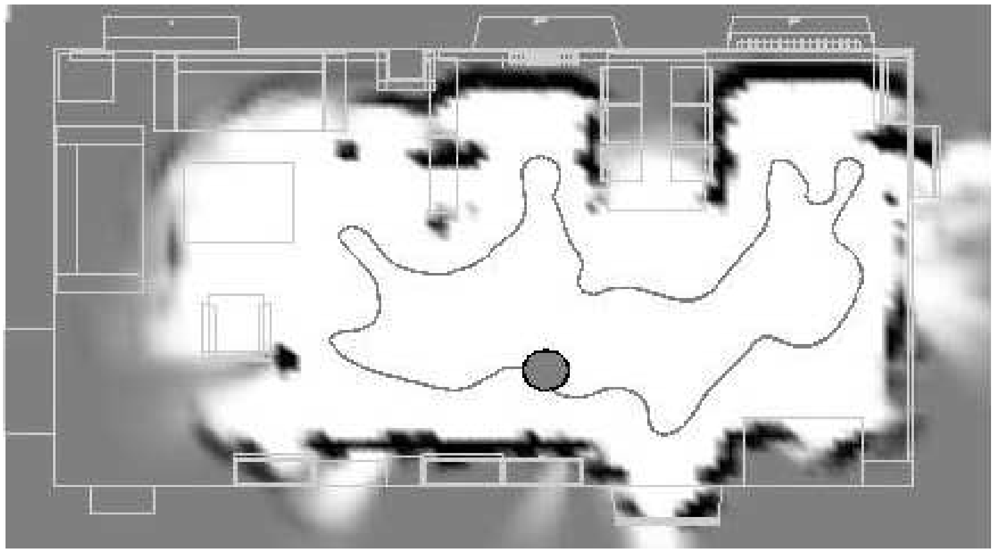
Summary Features Maps

- Pros:
  - Compact
  - High accuracy
  - Easy to edit/make by human
  - Can be generated from drawings without using the sensor
  - Can be shared across sensors
- Cons:
  - Sparse representation, only features seen, i.e. does not model for example free space
  - No to know what features to used a priori

Grid maps

- Divide the world into cells
- Map defines state of each cell (occupied or not)
- Called occupancy grids
- Was made popular in robotics by Moravec and Elfes[21]
- Several approaches to update the map (Bayes, Fuzzy, ...)

Example grid map



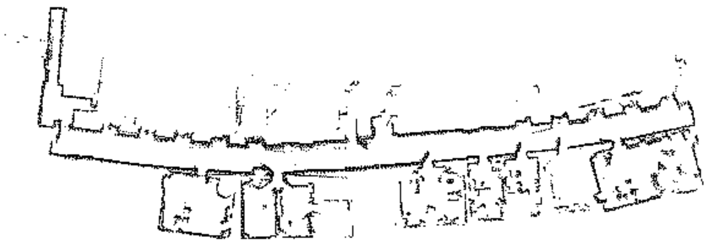
Summary Grid Maps

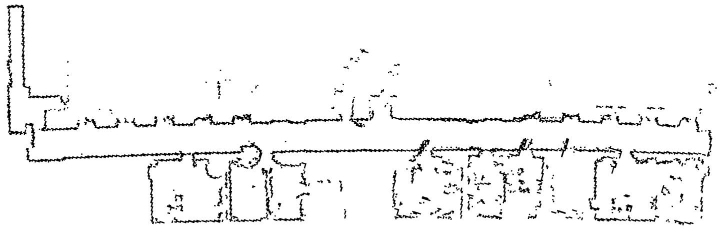
- Pros:
  - Model free[13]
  - Good for integration of data such as sonar[22]
  - Dense representation
  - Visually more complete
- Cons:
  - Not so easy to edit by human
  - Often computationally more intense to work with

Appearance Based Maps

- Uses raw or little processed data
- Motivation: Loosing information when extracting features or putting data in grid
- Stores reference data associated with a certain pose
- Localization achieved by match against reference data
- Often used with vision, sometimes laser scans[18, 19]
- PCA techniques also used
  - Laser data Wallner & Crowley[26, 7]
  - Images Sim & Dudek [25]

Scan Match Map



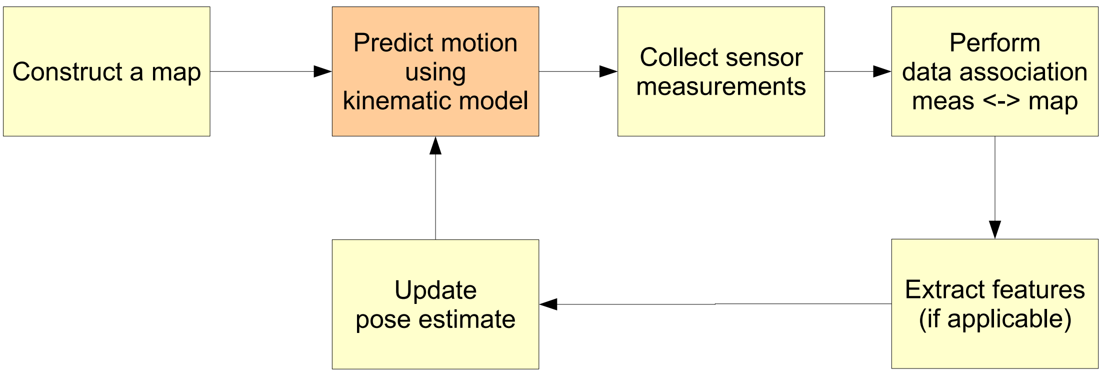


- 148 laser scans in groups of four
- Processed using Steffen Gutmann's ScanStudio

Hybrid Methods

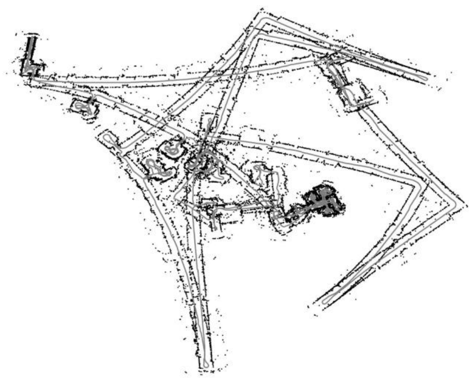
- No single represenation works in all situations
- Want to combine detail and scalability
- Hybrid maps getting more and more common
- Example: Combine one topological map with one feature map per room
- Get scalability from topological map and accuracy from metric maps

Prediction



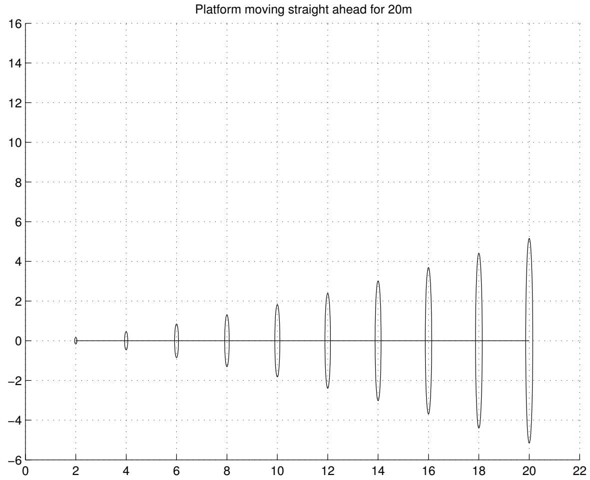
Wheel encoders

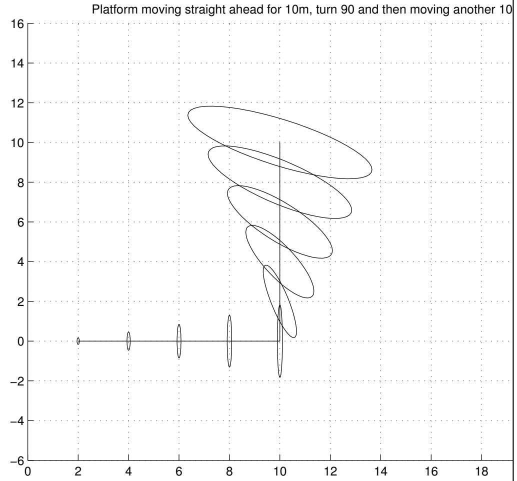
- Most robots have wheel encoders ⇒ odometry
- Provides information about relative motion
- Typically very accurate at short range
- Will drift over longer ranges
- Error in dead-reckoning unbounded
- Angular error ⇒ large position errors



Path Dependent Uncertainty

- The uncertainty that is accumulated depends on the path
- Typically quite accurate for translations
- Turning often increases uncertainty (slipping, . . .)



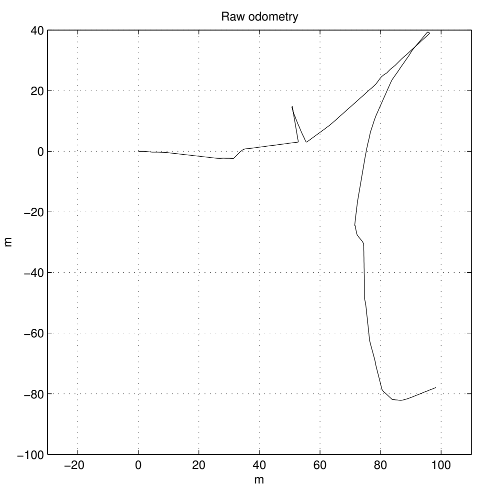


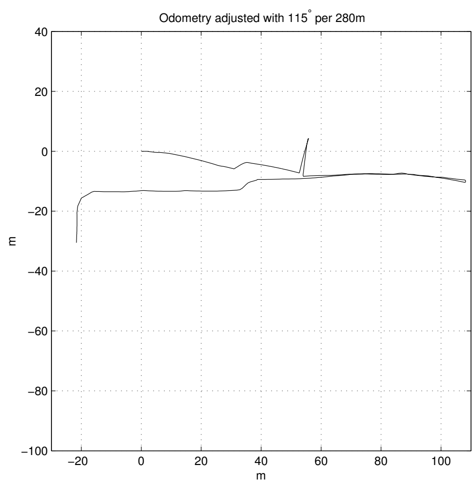
Sources of errors

- Systematic errors (some of them)
  - Wrong size of wheels in model
  - Misalignment of the wheels
  - Limited encoder resolution and sampling rate
  - ...
- Non-systematic errors
  - Uneven surface ⇒ wheels travel further
  - Objects on the floor ⇒ large disturbance
  - Wheel slippage (slippery floors, over-acceleration, fast-turning, external forces, non-point wheel contact with floor ...)

Compensating for systematic errors

- Systemic errors give bias which many estimation methods do not handle well
- Systematic errors can be reduced significantly by calibration[4, 5]
- Example: Adjust direction of the platform with 0.4°/m





Kinematic model

- Pose given by

$$\bar{x} = \begin{pmatrix} x & y & \theta \end{pmatrix}^T$$

- Motion of right and left wheel Δd_r and Δd_l
- Distance between the wheel B

$$\begin{aligned}
\Delta d &= \frac{\Delta d_r + \Delta d_l}{2} \\
\Delta \theta &= \frac{\Delta d_r - \Delta d_l}{B} \\
\Delta x &= \Delta d \cos(\theta + \frac{\Delta \theta}{2}) \\
\Delta y &= \Delta d \sin(\theta + \frac{\Delta \theta}{2})
\end{aligned}$$

Kinematic model

- In condensed form

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_{k+1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}_k + \begin{pmatrix} \Delta d \cos(\theta + \frac{\Delta\theta}{2}) \\ \Delta d \sin(\theta + \frac{\Delta\theta}{2}) \\ \Delta \theta \end{pmatrix}$$

- In general

$$x_{k+1} = f(x_k, u_k)$$

Uncertainty propogation

- Let P_k be the covariance matrix describing the uncertainty of x_k
- Uncertainty for x_{k+1} can be calculated based on

$$x_{k+1} = f(x_k, u_k)$$

- where $u = (\Delta d_r, \Delta d_l)^T$ is the system input
- The covariance of x_{k+1} if then given by

$$P_{k+1} := (\nabla_x f) P_k (\nabla_x f)^T + (\nabla_u f) Q (\nabla_u f)^T$$

- $\nabla_x f$ is the Jacobian of $f(x_k, u)$ w.r.t. x
- $\nabla_u f$ is the Jacobian of $f(x_k, u)$ w.r.t. u
- Compare end of last lecture

Uncertainty propogation cont'd

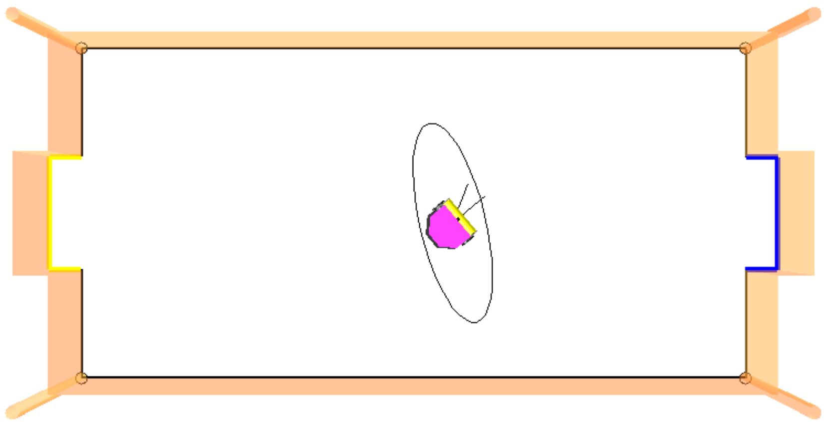
- Common model for Q

$$Q = \begin{pmatrix} k_r|\Delta d_r| & 0 \\ 0 & k_l|\Delta d_l| \end{pmatrix}$$

- Model assumes uncertainty is propotional to distance travelled
- k_i chosen to given reasonable uncertainty

Example: Prediction error

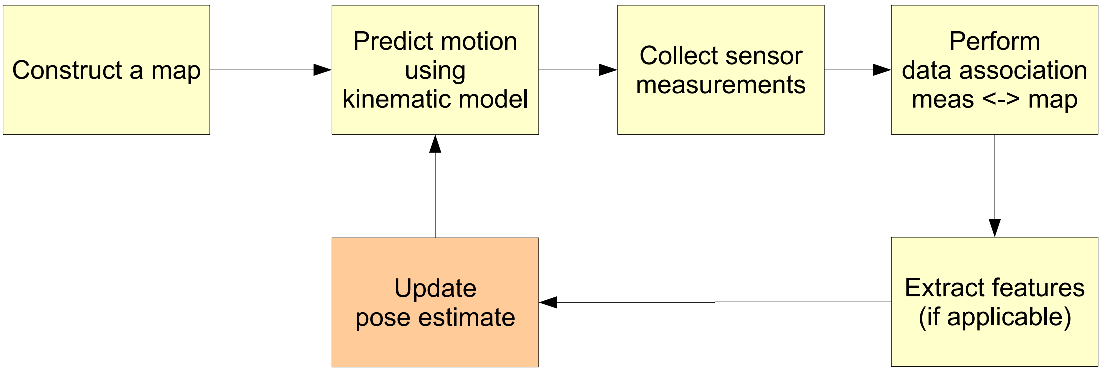
- Test program under /usr/local/pkg/localizationdemo
- Start ./ekfLocalization -P
- Investigate different noise levels and parameters for odometry (press 's' to change values)



Comments to prediction step

- Often need to make the prediction quite pessimistic (k_i large in the model for Q)
- Need to account for large unforeseen errors
  - slippage, driving over cables, etc
- Important to be able to detect such events to avoid getting lost

Pose update



Kalman Filter

- Much of the work in mapping and localization is built on the Kalman filter[17]
- Described in many books[12, 20, 2]
- An introduction can be found at http://www.cs.unc.edu/~ welch/kalman
- It is an optimal recursive data processing algorithm[20]
- Recursive $\Rightarrow$ can be used online

System model

- Most systems have some non-linearity either in motion model or in the model that describes the measurements

$$
\begin{aligned}
x_k &= f(x_{k-1}, u_k, w_k) && \text{(process model)} \\
z_k &= h(x_k, v_k) && \text{(measurement model)}
\end{aligned}
$$

- Cannot apply Kalman filter as is because it assumes the model is linear.
- Need to make approximation
- Use Extended Kalman Filter (EKF)

Extended Kalman Filter

- Prediction step:

$$
\begin{aligned}
\hat{x}_{k|k-1} &= f(\hat{x}_{k-1|k-1}, u_k, 0) \\
P_{k|k-1} &= F_k P_{k|k-1} F_k^T + G_k Q_k G_k^T.
\end{aligned}
$$

- Update step:

$$
\begin{aligned}
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_k, 0)) \\
P_{k|k} &= P_{k|k-1} - K_k H_k P_{k|k-1}
\end{aligned}
$$

where

$$
\begin{aligned}
S_k &= H_k P_{k|k-1} H_k^T + R_k \\
K_k &= P_{k|k-1} H_k^T S_k^{-1}
\end{aligned}
$$

Notation

- $\hat{x}_{k|k-1}$: estimate of x_k using measurements up to time $k-1$
- $\hat{x}_{k|k}$: estimate of x_k using measurements up to time k
- P: covariance matrix of state estimate
- Q: process noise covariance matrix
- R: measurement noise covariance matrix

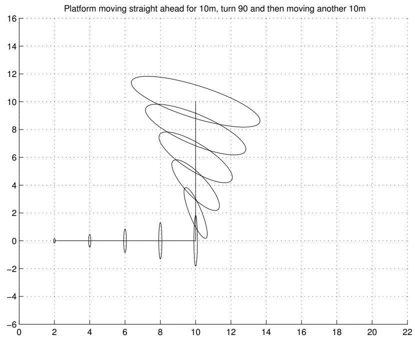
Notation cont'd

- S: measurement innovation covariance matrix
- K: Kalman gain
- F_k: Jacobian of f w.r.t. x, i.e. $F_k = \nabla f_x(\hat{x}_{k|k-1}, u_k, 0)$
- G_k: Jacobian of f w.r.t. w, i.e. $G_k = \nabla f_w(\hat{x}_{k|k-1}, u_k, 0)$
- H_k: Jacobian of h w.r.t. x, i.e. $H_k = \nabla h_x(\hat{x}_{k|k-1}, 0)$

Closer Look at Prediction Step

$$
\begin{aligned}
x_{k|k-1} &= f(x_{k-1|k-1}, u_k, 0) \\
P_{k|k-1} &= F_k P_{k|k-1} F_k^T + G_k Q_k G_k^T .
\end{aligned}
$$

- Use zero mean assumption on noise and set $w = 0$
- Increases the uncertainty in the state estimate
  $\Rightarrow$ P increases
- Good process model f important (motion model of robot)



Closer Look at Update Step

$$
\begin{aligned}
S_k &= H_k P_{k|k-1} H_k^T + R_k \\
K_k &= P_{k|k-1} H_k^T S_k^{-1} \\
x_{k|k} &= x_{k|k-1} + K_k(z_k - h(x_k, 0)) \\
P_{k|k} &= P_{k|k-1} - K_k H_k P_{k|k-1}
\end{aligned}
$$

- Use zero mean assumption on noise and set $v = 0$ in h
- Reduces uncertainty in state estimate
  $\Rightarrow$ P decreases

Behaviour

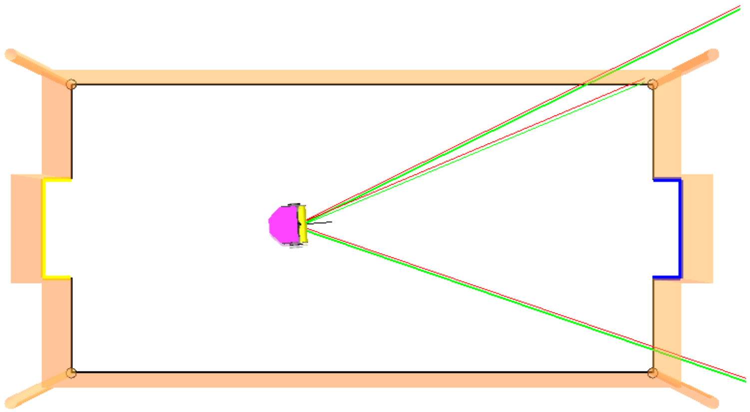
- Small mesaurement noise level
  $\Rightarrow$ Not much filtering. Estimate follows measurement closely (not smooth)
- Small process noise level $\Rightarrow$ Estimate not effected as much, estimate smoothed based on process model (motion model for robot)
- It is the relative size of the process and measurement noise that matters for the estimate of the state, so increasing process noise has the same effect as decreasing the measurement noise and vice versa.

Localization with EKF

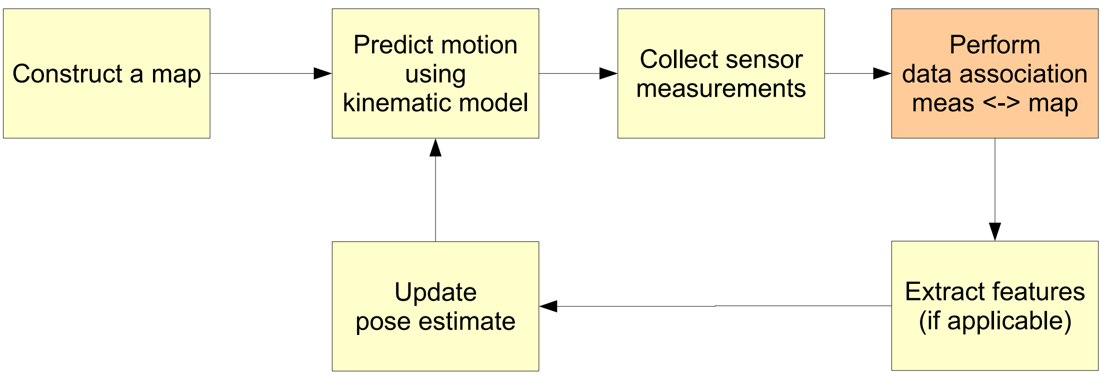
- State vector: $x = [x, y, \theta]^T$
- Use measurements to update the estimate $\hat{x}$ of the robot pose
- Formal: Estimate $p(\hat{x}|Z^k, \mathcal{M})$
- Measurement history:
  $Z^k = \{z_0, z_1, \ldots, z_k\}$
- Map:
  $\mathcal{M} = \{m_0, m_1, \ldots, m_M\}$

Example: EKF Localization

- Run ekfLocalizer under /usr/local/pkr/localizationdemo
- Play with parameter settings
- Investigate bearing-only measurements vs range+bearing

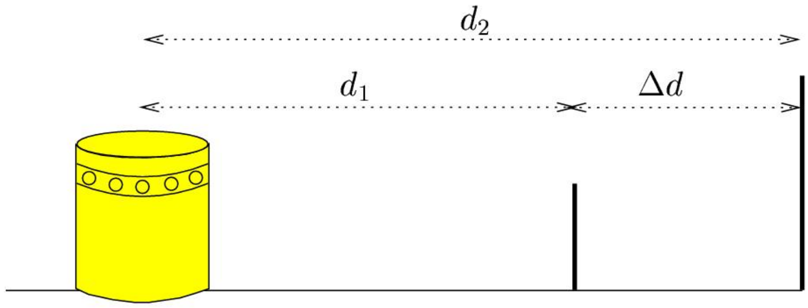


Data association



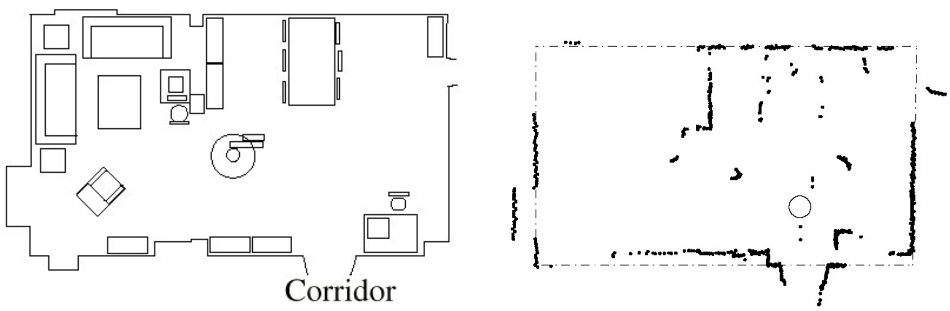
Data Association

- Correspondence problem
- One of the hardest problems in robotics
- What in our model does the measurement correspond to?



Example: Data Association

- What points in the laser scan belong to the walls?



Data Association

- The *innovation* is defined as $\nu_k = z_k - h(\hat{x}, 0)$
- Gives the difference between what we measure and what we predict to measure ($\hat{z}_k = h(\hat{x}, 0)$).
- Natural to look at the innovation to tell if we have a correct measurement
- Idea: "If innovation too large skip the measurement"
- The problem with this is that that "too large" is hard to define with a fixed threshold
- It depends on the uncertainty!
- When we are unsure of the state we have to accept larger innvations and vice verse

Mahalanobis Distance

- Close often measured by mahalanobis distance

$$\rho_{i,j} = \nu_{i,j} S_{i,j}^{-1} \nu_{i,j}^T$$

  where $\nu_{i,j}$ is the innovation given when associating measurement i with map entity j and $S_{i,j}$ is the corresponding measurement covariance matrix
- Weights the innovation with the uncertainty in the model and the measurement
- Small innovation required if uncertainties are small (S small)
- and vice versa

Mahalanobis Distance cont'd

- The Mahalanobis distributed according to a χ^2-distribution[3]
- Can use hypothesis testing when looking for matches
- Null-hypothesis: The measurement matches
- Reject null-hypothesis on significance level α if

$$\rho > \chi^2_{\alpha,n}$$

where n is the dimension of ν.

Hypothesis Test cont'd

- $\chi^2_{\alpha,n}$ given by tables

n/Q	0.10	0.05	0.025	0.01	0.001
1	2.706	3.841	5.024	6.635	10.828
2	4.605	5.991	7.378	9.210	13.816
3	6.251	7.815	9.348	11.345	16.266

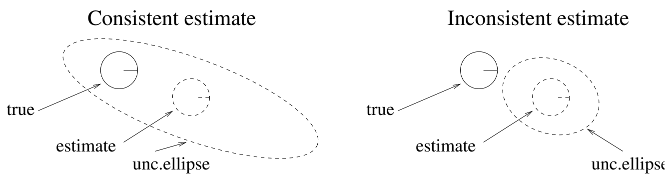
- In this case $\alpha = 1 - Q$
- Example: A measurement with two variables ($d = 2$) is reject as a match if $\rho > 5.991$ on significance level 0.95.
- This means that we will falsely reject 5% of the true matches
- Trade-off between false negative and false positives
- Errors in data association can be "fatal"

Nearest Neighbor Matching

- Ideally a measurement matches only one model feature
- Nearest Neighbor (NN) is common approach in cases of multiple matches
- Pick the one with the smallest mahalanobis distance[3]
- Some choose to reject the measurement when there is an ambiguity

Consistency

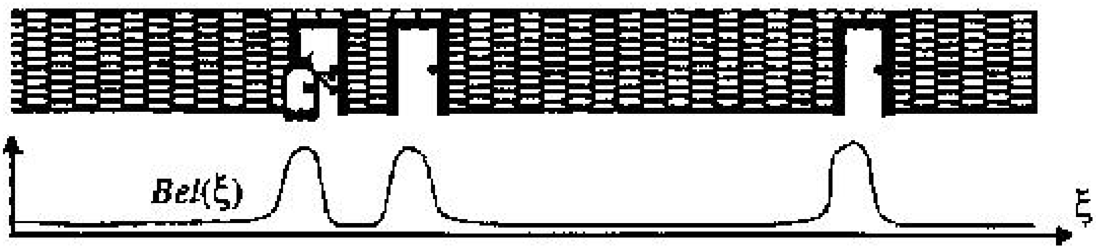
- The state covariance matrix P gives information about the uncertainty
- Consisteny: The uncertainty captures the real difference between the estimate and the true state
- Errors in data association can give inconsistency
- Uncertainty is reduced due to update but estimate no longer inside uncertainty ellipse



What if the robot cannot make up its mind?

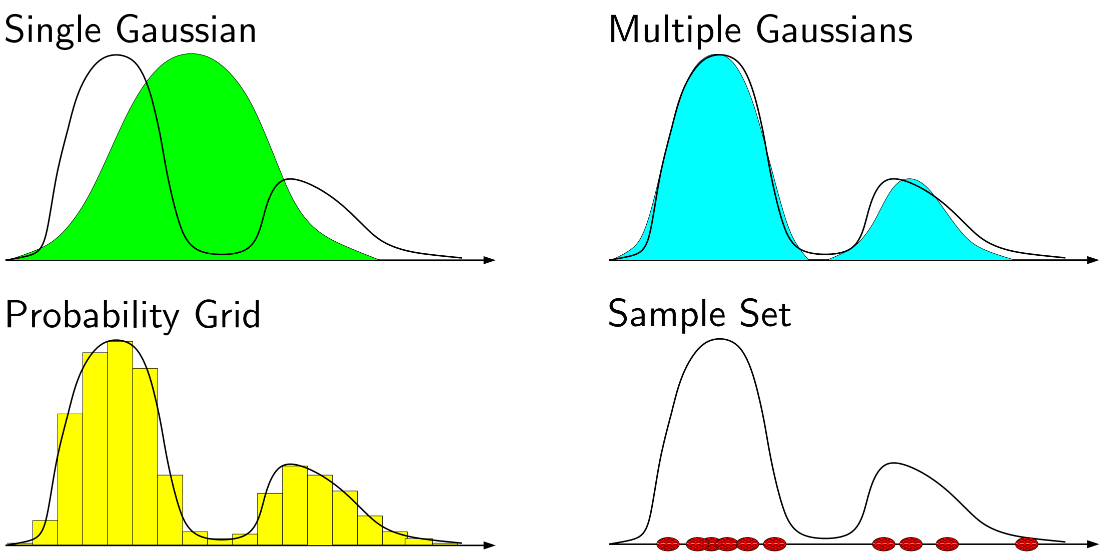
- Too large uncertainty for reliable data association
- Initialization

⇒ Multi-modal distribution is needed



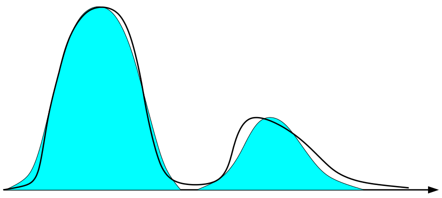
Picture from [11]

Different representations



Multiple Gaussians

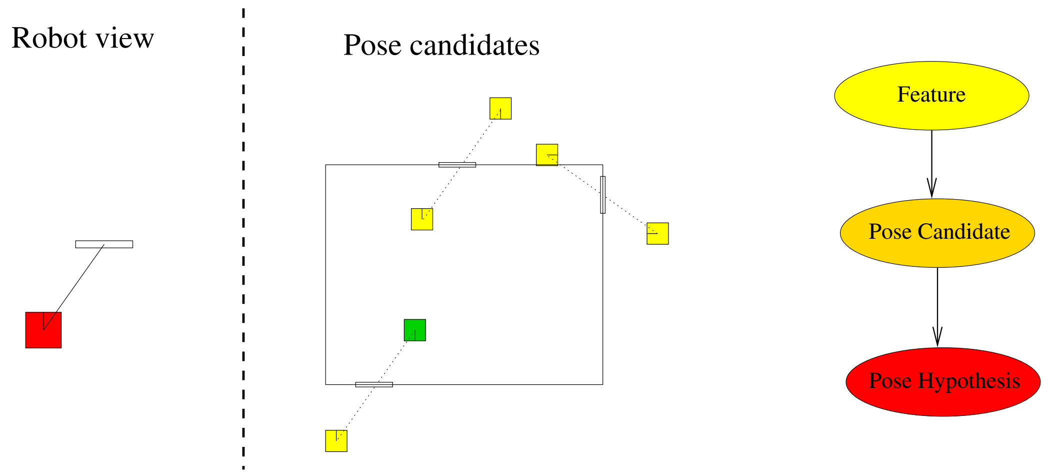
- Jensfelt 99[16], Roumeliotis 00[24], Reuter 00[23], . . .
- Natural extension
- Each Hypothesis a Gaussian



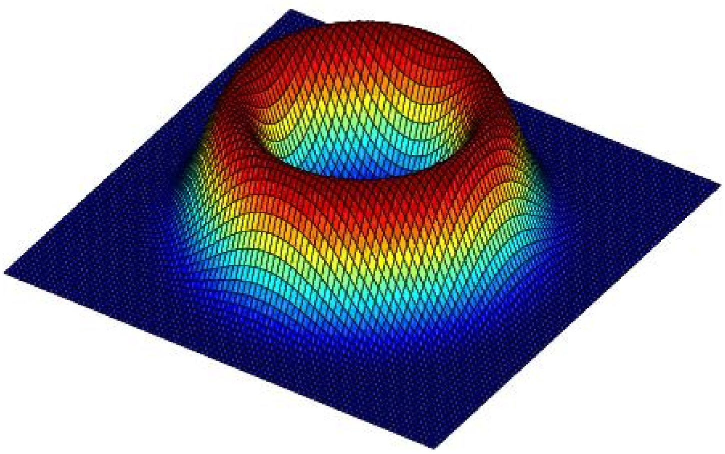
Now we can track multiple hypotheses!

- Data association problem resolved by creating new hypothesis
- Non-trivial to determine if we have a match
- Explosion of hypotheses
- Must remove unlikely hypotheses

Global Localization



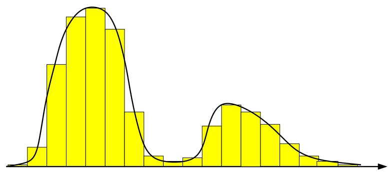
But what about single point landmarks?



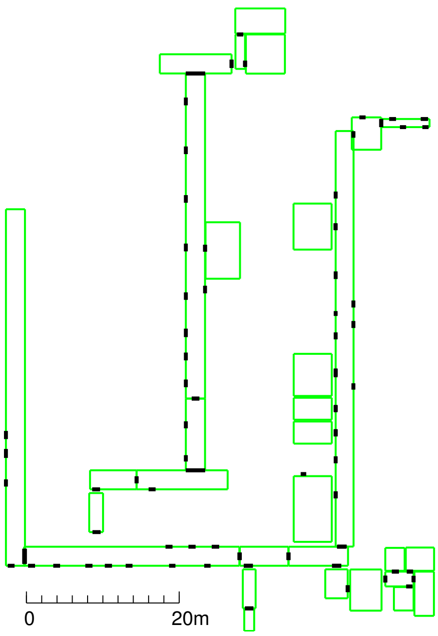
Gaussian assumption too strong!

Position Probability Grids

- Burgard 96[6], Fox 98[10], . . .
- Discretize the PDF by dividing the world into a grid
- Each cell represents on possible pose (one state)
- Handles multi-modal PDFs
- No assumption like the Gauss
- But ...



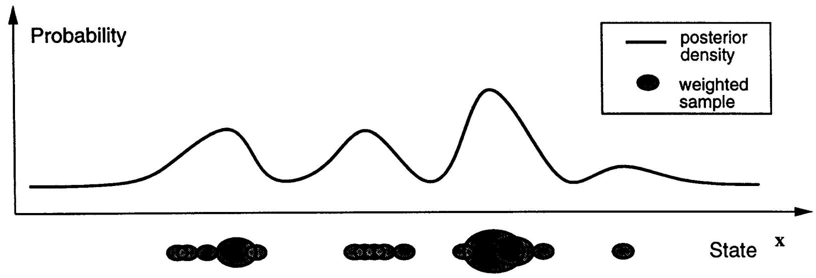
CPU will overheat!



- 900 m^2
- 0.1x0.1 m^2 and 5° cells
- $\Rightarrow \approx 10^6$ cells
- In theory each cells has to be updated for each measurement
- Approximations needed to run in almost real-time
- Many cells with $p(x|\mathcal{F}) \approx 0$
- Variable cell size one idea

Particle Filter

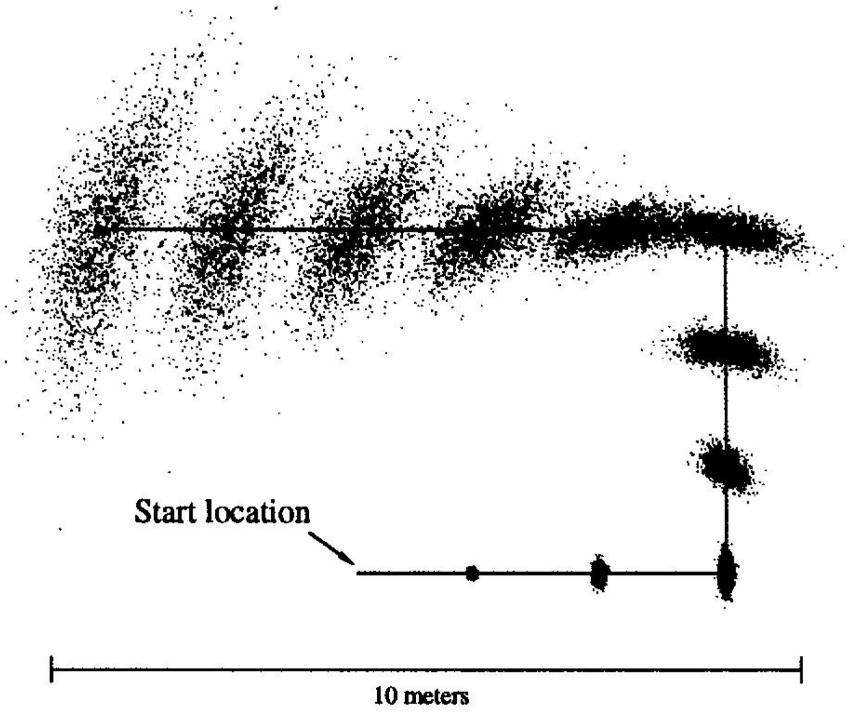
- Dellaert 99[8], Fox 99[9], Jensfelt 00[15] ...
- Sample the PDF
- Each sample/particle has a pose, s_k, and a weight, π_k



Particle Filter cont'd

- Maintain and update the set of samples based on:
  - Predictions $\Rightarrow$ Diffusion, i.e. particles spread
  - Observations $\Rightarrow$ Clustering of particles

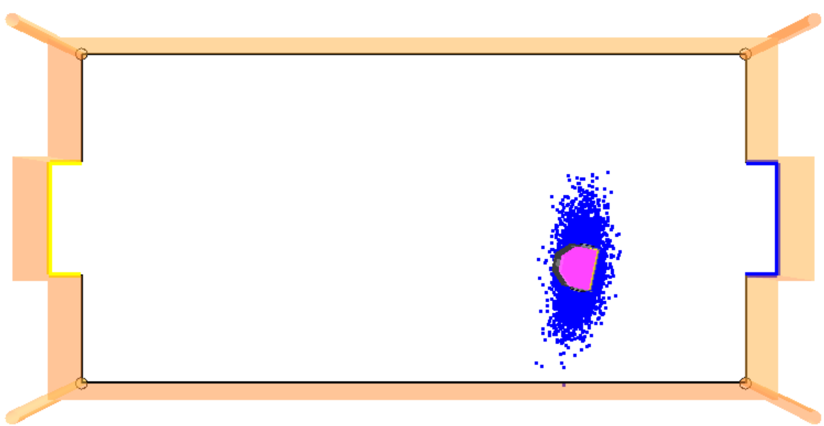
Illustration



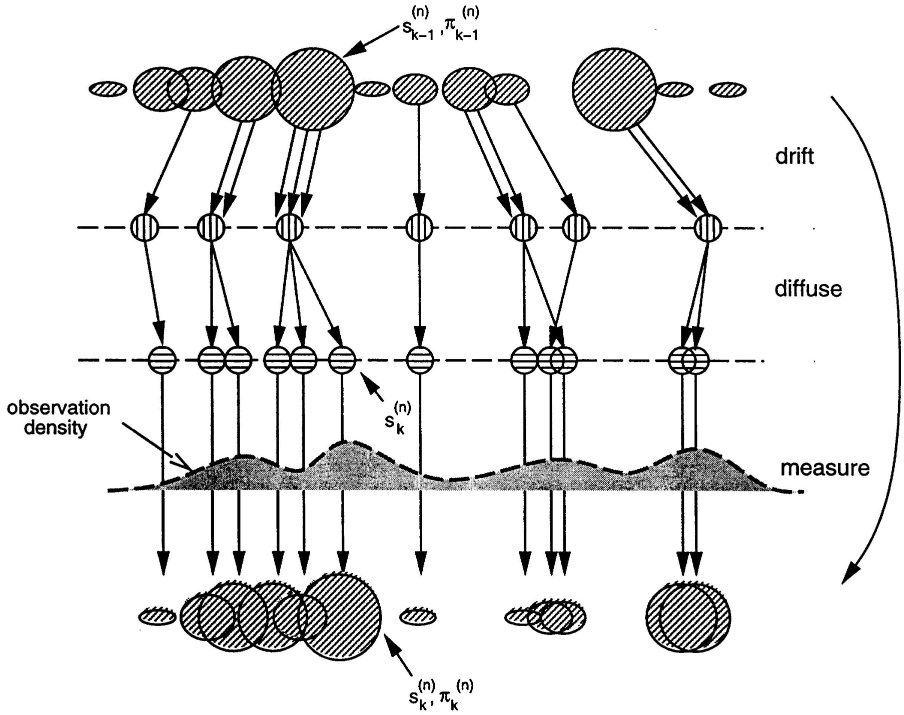
Picture from [11]

Example: Particle filter

- Go to /usr/local/pkg/localizationdemo
- Run ./particleLocalizer -P



The Algorithm

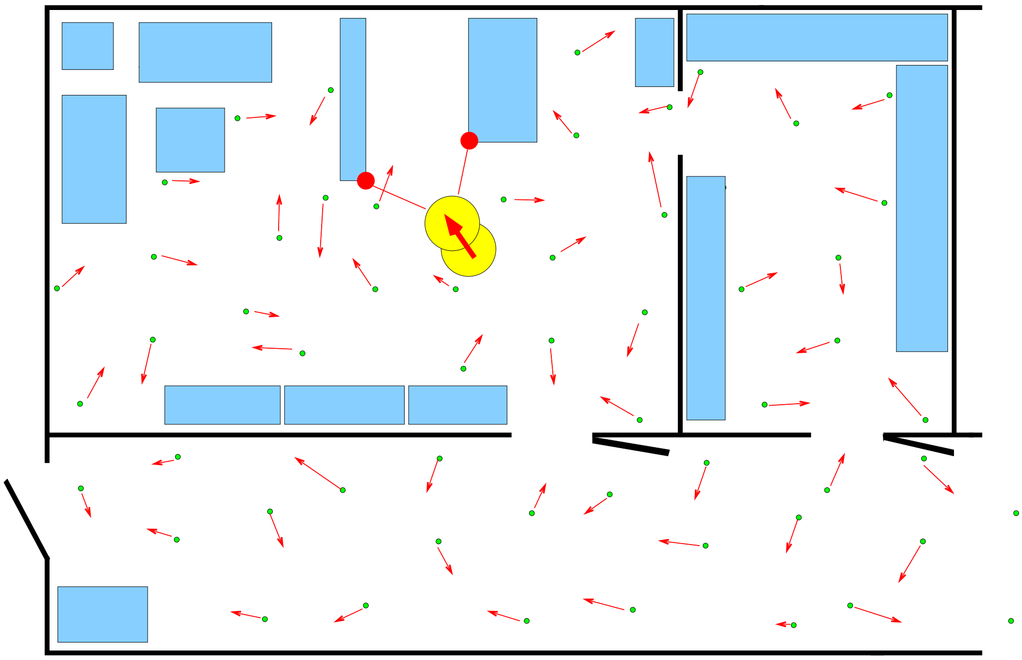


Picture from [14]

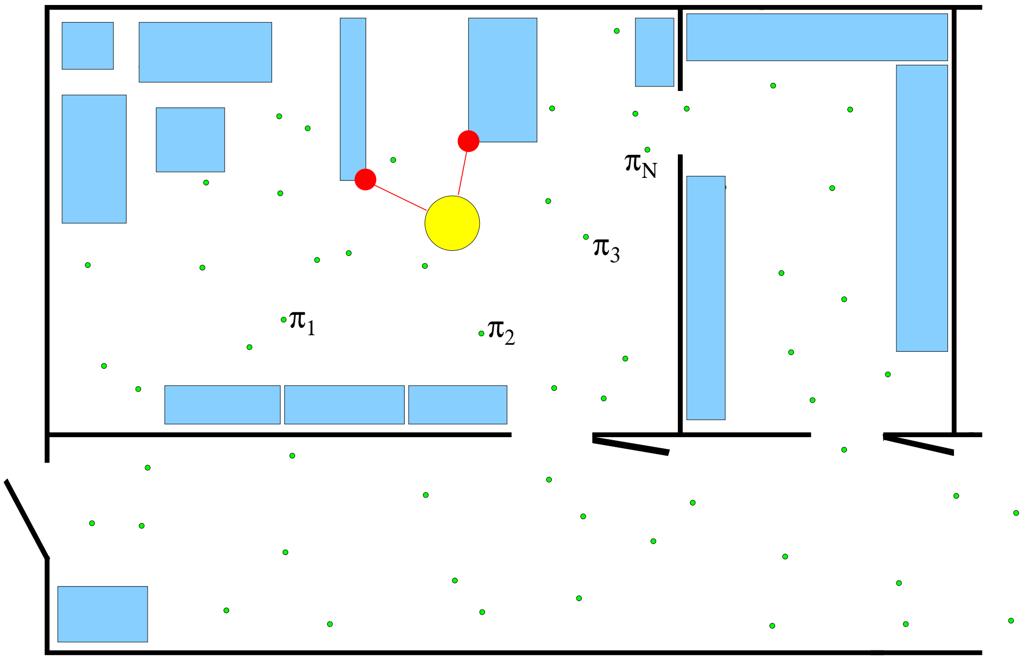
Observations

- Particles with high weight ⇒ attract more particles
  survival of the fittest!
- Weak particles ⇒ likely to disappear
  use particles where they are needed
- Need initial particles were PDF is significant
- Cannot create new particles, stuck with initial set

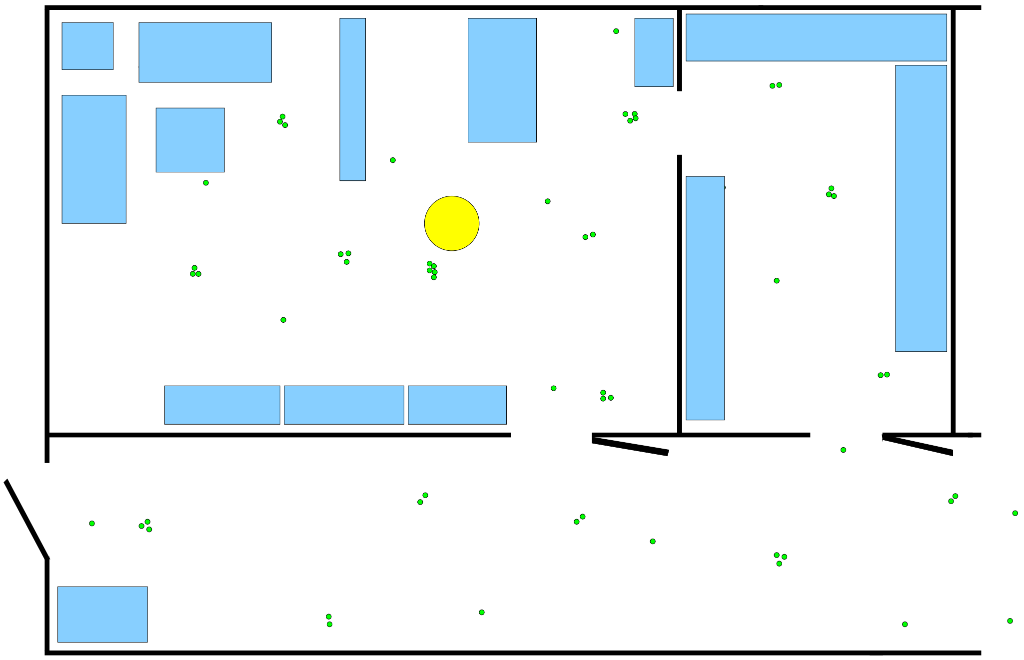
Predict and Diffuse



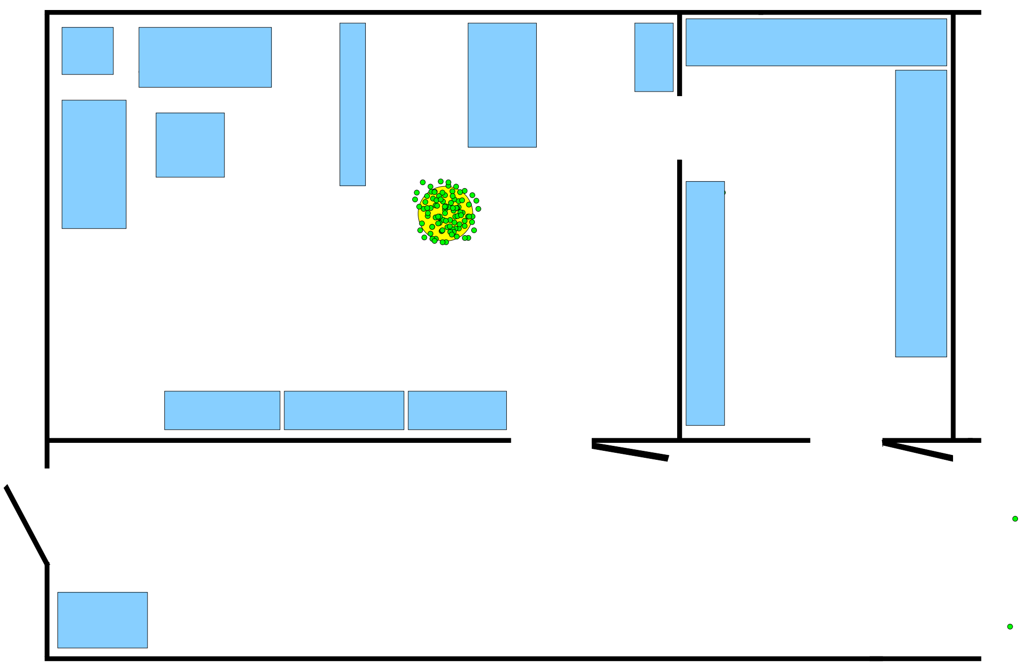
Measurement update



Re-sampling

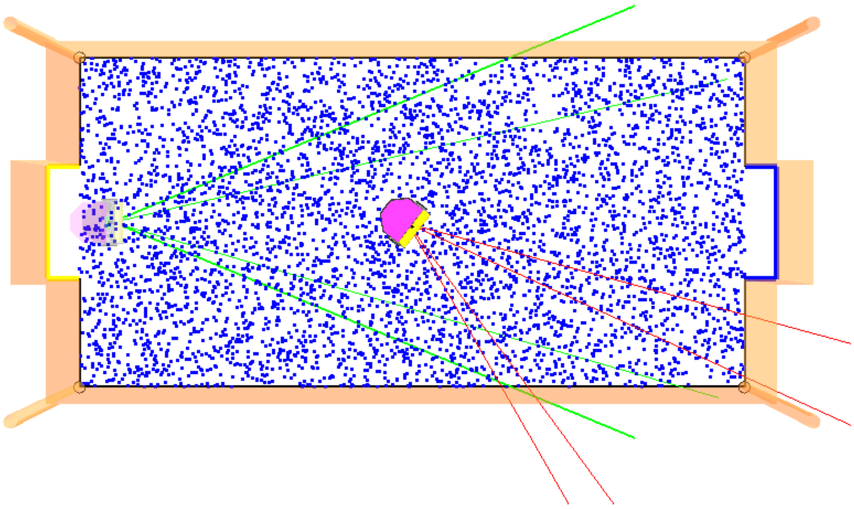


YES!!!!!!!!!!!!!!!



Example: Particle filter

- Go to /usr/local/pkg/localizationdemo
- Run ./particleLocalizer



SLAM

- Most of the research has shifted away from localization
- SLAM attracts much more attention
- Especially visual SLAM

Brief on SLAM

- Can use simular metods to those used in localization
- With EKF extend the state to include the position of the features in the map as well.
- Write measurement function as function of the feature parameters as well
- EKF SLAM does not scale very good $O(N^2)$
- Has been the motivation for much of the current research

SLAM summer school 2002, http://www.cas.kth.se/slam/, August 2002.

B. D. O. Anderson and J. B. Moore.
Optimal Filtering.
Prentice-Hall, Englewood Cliffs, N. J., 1979.

Yaakov Bar-Shalom and Thomas E. Fortmann.
Tracking and Data Association.
Academic Press, Inc., 1988.

J. Borenstein and L. Feng.
Umbmark - a method for measuring, comparing, and correcting dead-reckoning errors in mobile robots.
Technical Report UM-MEAM-94-22, Department of Machanical Engineering and Applied Mechanics, The University of Michigan, dec 1994.

Johann Borenstein and Liqiang Feng.
Correction of systematic odometry errors in mobile robots.

In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95)*, pages 569–574, Pittsburgh, PA, August 5-9, 1995. IEEE/RSJ.

Wolfram Burgard, Dieter Fox, Daniel Henning, and Timo Schmidt.
Estimating the absolute position of a mobile robot using position probability grids.
In *Proc. of the National Conference on Artificial Intelligence (AAAI-96)*, pages 896–901, Portland, Oregon, USA, August 1996.

James L. Crowley, Frank Wallner, and Bernt Schiele.
Position estimation using principal components of range data.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 3121–3128, Leuven, Belgium, May 1998. IEEE.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun.
Monte carlo localization for mobile robots.

In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)*, volume 2, pages 1322 – 1328, Detroit, Michigan, USA, May 10-15, 1999.

Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun.
Monte carlo localization - efficient position estimation for mobile robots.
In *Proc. of the National Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida, USA, July 18-22, 1999.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun.
Active markov localization for mobile robots.
Robotics and Autonomous Systems, 25:195–207, 1998.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun.
Markov localization for mobile robots in dynamic environments.
Journal of Artificial Intelligence Research, 11:391–427, 1999.

A. Gelb, editor.
Applied Optimal Estimation.

MIT Press, 1974.

G. Hager and M. Mintz.
Sensor modeling and robust sensor data fusion.
In *Proc. of the International Symposium on Robotics Research*, pages 69–74, 1990.

Michael Isard and Andrew Blake.
CONDENSATION – conditional density propagation for visual trackoing.
International Journal on Computer Vision, 29(1):5–28, 1998.

Patric Jensfelt, David Austin, Olle Wijk, and Magnus Andersson.
Feature based condensation for mobile robot localization.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)*, volume 3, San Francisco, CA, USA, May 2000.

Patric Jensfelt and Steen Kristensen.
Active global localisation for a mobile robot using multiple hypothesis tracking.

In *Proc. of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation (avail at http://www.dsv.su.se/ijcai-99/)*, pages 13–22, Stockholm, Sweden, August 1999.

R.E. Kalman.
A new approach to linear filtering and prediction problems.
Transactions of the ASME, 82:35–45, March 1960.

Feng Lu and E.E. Milios.
Robot pose estimation in unknown environments by matching 2d range scans.
In *Proc. of 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pages 935–938, June 1994.

Feng Lu and Evangelos Milios.
Optimal global pose estimation for consistent sensor data registration.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'95)*, pages 93–100, 1995.

Peter S. Maybeck.
Stochastic models, estimation and control, volume 1.
Academic Press, 1979.

H.P. Moravec and A. Elfes.
High resolution maps from wide angle sonar.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)*, pages 116–121. IEEE, 1985.

U. Raschke and J. Borenstein.
A comparison of grid-type map-building techniques by index of performance.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'90)*, volume 3, pages 1828–1832. IEEE, 1990.

Johannes Reuter.
Mobile robot self-localization using pdab.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)*, volume 4, pages 3512–3518. IEEE, April 2000.

Stergios I. Roumeliotis and George A. Bekey.
Bayesian estimation and kalman filtering: A unified framework for mobile robot localization.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)*, volume 3, pages 2985–2992. IEEE, April 2000.

Robert Sim and Gregory Dudek.
Learning visual landmarks for pose estimation.
In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)*, volume 3, pages 1972–1978, 1999.

Frank Wallner.
Position Estimation for a Mobile Robot from Principal Components of Laseer Data.
PhD thesis, INPG, October 1997.