Robótica Móvel

Localization

Part 3 - Probabilistic Positioning

Universidade de Aveiro
Vitor Santos, Nuno Lau — 2023-2024

Summary

References and further reading

- Webster, J.G., Huang, S. and Dissanayake, G. (2016). Robot Localization: An Introduction. In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (Ed.). https://doi.org/10.1002/047134608X.W8318
- RAM and RMI courses at the University of Aveiro

Introduction

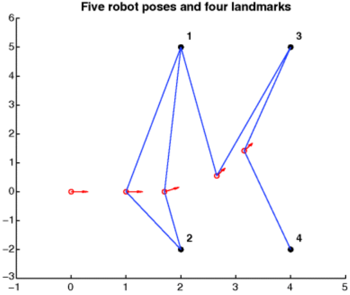
Introduction

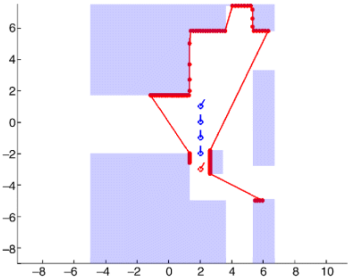
- Integration of motion measurements has incremental errors and uncertainties and detection of external features also have uncertainties
- Probabilistic techniques are used to improve the estimation of the localization by using models of motion and perception
  - Kalman filtering
  - Particle filter (or Monte Carlo) localization
  - Markov localization

Models

- To perform localization estimation two types of models are required:
  - Robot motion model
  - Sensor model (also known as measurement or observation model)
- Motion models are the kinematics equations specific of each type of robot tipoplogy
- Sensor model
  - Relationship between the observations from the sensors and the location of the robot in the map.
  - Depends on the sensor characteristics and on the way the map is represented
- The map of the environment is typically defined by:
  - Coordinates of known landmarks or features
  - Occupancy grid with cells being occupied or free.

Examples of Maps





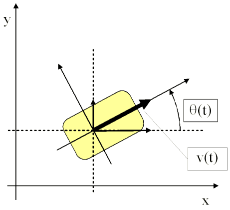
- Localization in a landmark-based map.
- The map is defined by four landmarks
- The robot starts from $(0,0,0)^\intercal$.
- At time steps 1 and 2, it observes landmarks 1 and 2; etc.
- Robot to landmark observations are indicated by blue lines.

- Localization with an occupancy grid map;
- Shaded areas represent occupied cells;
- White area represents the free space;
- The robot moves a few steps;
- Readings from LiDAR at the first pose depicted as red lines.

Vehicle model (with noise)

- Consider the generic kinematic equations of a robot in the plane

$$\begin{cases} \dot{x}(t) = (V(t) + \delta V(t)) \cos\theta(t) \\ \dot{y}(t) = (V(t) + \delta V(t)) \sin\theta(t) \\ \dot{\theta}(t) = \omega(t) + \delta\omega(t) \end{cases} \quad (1)$$



- with $\delta V(t)$ and $\delta\omega(t)$ the differences between the intended and the actual control values (control noises); assumed zero-mean Gaussian.
- After integration (inverse kinematics, as seen earlier), we can establish the discrete version with a sampling time Δt:

$$\begin{cases} x_{k+1} = x_k + (V_k + \delta V_k)\Delta t \cos\theta_k \\ y_{k+1} = y_k + (V_k + \delta V_k)\Delta t \sin\theta_k \\ \theta_{k+1} = \theta_k + (\omega_k + \delta\omega_k)\Delta t \end{cases} \quad (2)$$

- The position variables at iteration $k+1$ are calculated after the values from iteration k including the velocity noises (δV_k, $\delta\omega_k$).

Sensor Model for Landmark-Based maps (with noise)

- Consider an environment with N known landmarks at positions (x_L^i, y_L^i), $i = 1, \cdots, N$
- We consider a sensor able to measure both range r_{k+1}^i and heading ϕ_{k+1}^i for iteration $k+1$ for landmark i (observation model):

$$\begin{cases} r_{k+1}^i = \sqrt{\left(x_L^i - x_{k+1}\right)^2 + \left(y_L^i - y_{k+1}\right)^2} + w_{\mathsf{r}} \\ \phi_{k+1}^i = \arctan \frac{y_L^i - y_{k+1}}{x_L^i - x_{k+1}} - \theta_{k+1} + w_\phi \end{cases} \tag{3}$$

- where w_{r} and w_ϕ are zero-mean Gaussian observation noises.
- Notes:
  - In case the sensor only observes one of these quantities, only the respective equation defines the sensor model.
  - LiDAR and Ultrasonic sensors usually provide both distance and bearing, but cameras normally provide only bearing!
  - Knowledge of the landmarks positions (x_L^i, y_L^i) is assumed to be precise without noise, although it is possible to expand the model also to include that situation.

Sensor Model for Occupancy Grid maps (with noise)

- Occupancy grid maps are a discretized representation of an environment with grid cells classified as occupied or free.
- A sensor on the robot can determine the distance to the nearest occupied cell along a given direction (LiDAR can do it)
- If there is no obstacle within the sensor range, the sensor typically reports a nominal maximum distance d_{max}.
- Although range measurements depend on the environment and robot location, it is not feasible to find an analytical observation model of the same form as for Landmark-based maps.
- However, given an estimate of the robot location and the grid map, the expected value of a range measurement can be numerically obtained using ray casting.
- This makes it possible to evaluate the likelihood of a given pose, which is sufficient in some localization approaches such as a particle filter.

Formulation of the problem in landmark-based maps

- The localization problem in a landmark-based map is to find the robot pose at time $k+1$:
  - $\mathbf{x}_{k+1} = \begin{bmatrix} x_{k+1} & y_{k+1} & \theta_{k+1} \end{bmatrix}^\mathsf{T}$
- given:
  - the map (set of landmarks)
  - the sequence of robot actions V_i, ω_i $(i = 0, \cdots, k)$
  - and sensor observations from time 1 to time $k+1$
- In its most fundamental form, the problem is to estimate the robot poses $\mathbf{x}_i$ $(i = 0, \cdots, k+1)$ that best agree with all robot actions and all sensor observations.
- This can be formulated as a nonlinear least-squares problem using the motion and observation models derived earlier – eq. (2) and (3).
  - The solution to the resulting optimization problem can then be calculated using an iterative scheme such as Gauss–Newton to obtain the robot trajectory and as a consequence the current robot pose.
  - However, due to the dimensionality of the problem and given the high sampling rate of modern sensors, this strategy quickly becomes computationally intractable!

Extended Kalman Filter for Localization

Extended Kalman Filter for Localization in Landmark maps

- Assume sensor measurement noises to have a Gaussian distribution
- Initial estimate of robot location also Gaussian: $\mathbf{x}_0 \sim \mathcal{N}\left(\hat{\mathbf{x}}_0, P_0\right)$
  - $\hat{\mathbf{x}}_0$ is the estimated pose (state) at time 0
  - P_0 is the variance of the Gaussian distribution associated to this estimation.
  - As this is a multivariate Gaussian distribution, P_0 is actually a covariance matrix.
- An approximate solution to this nonlinear least-squares problem can be obtained using an Extended Kalman Filter (EKF)
- EKF effectively summarizes all the measurements obtained in the past in the estimate of the current robot location and its covariance matrix.
- A new observation from the sensor allows new estimates of the current robot location and its covariance.

EKF Formulation

- Being:
  - $\mathbf{u}_k = \begin{bmatrix} V_k \\ \omega_k \end{bmatrix}$ and $\mathbf{w}_k = \begin{bmatrix} \delta V \\ \delta\omega \end{bmatrix}$
- The pose estimation can be written as:
  - $\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right)$
- where:
  - f is the system transition function,
  - $\mathbf{u}_k$ is the control (input velocities),
  - and $\mathbf{w}_k$ is the zero-mean Gaussian process noise $\mathbf{w}_k \sim \mathcal{N}\left(0, Q\right)$.
- For the general case of more than one landmark observed we consider:
  - all observations r^i_{k+1} and ϕ^i_{k+1} together as a single vector $\mathbf{z}_{k+1}$,
  - and all the noises w_{r}, w_ϕ together as a single vector $\mathbf{v}_{k+1}$.
- The observation model at time $k+1$ as stated in eq. (3) can also be written in a compact form as:
  - $\mathbf{z}_{k+1} = h\left(\mathbf{x}_{k+1}\right) + \mathbf{v}_{k+1}$
- where
  - h is the observation function obtained from equation (3) and
  - $\mathbf{v}_{k+1}$ is the zero-mean Gaussian observation noise $\mathbf{v}_{k+1} \sim \mathcal{N}\left(0, R\right)$

Applying the EKF

- The localization problem is then to estimate $\mathbf{x}_{k+1}$ at time $k+1$:
  - $\mathbf{x}_{k+1} \sim \mathcal{N}\left(\hat{\mathbf{x}}_{k+1}, P_{k+1}\right)$
- where
  - $\hat{\mathbf{x}}_{k+1}$ and P_{k+1} are updated using the information from the sensors.
- The EKF framework uses the following steps:
  - Prediction using the process model
  - Update using observation
  - Perform a recursive application of the equations every instant a new observation is gathered
- This process yields an updated estimate for the current robot location and its uncertainty.
- This recursive nature makes EKF the most computationally efficient algorithm available for robot localization.

Equations to implement the EKF - Prediction

- Prediction using process model:

$$\begin{cases} \bar{\mathbf{x}}_{k+1} = f\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right) \\ \bar{P}_{k+1} = J_{f_x}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right) P_k J_{f_x}^T\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right) + \\ \qquad\qquad + J_{f_w}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right) Q_k J_{f_w}^T\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right) \end{cases} \tag{4}$$

- where
  - $J_{f_x}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right)$ is the Jacobian of function f with respect to $\mathbf{x}$;
  - $J_{f_w}\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right)$ is the Jacobian of function f with respect to $\mathbf{w}$;
  - both evaluated at $\left(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0\right)$
- Note: the value used for $\mathbf{w}_k$ is zero because that variable was considered earlier as zero-mean Gaussian with covariance matrix Q.

Equations to implement the EKF - Update

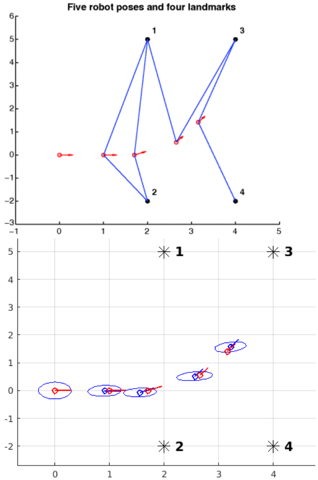
- Update using observation:

$$\begin{cases} \hat{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_{k+1} + K\left(\mathbf{z}_{k+1} - h\left(\bar{\mathbf{x}}_{k+1}\right)\right) \\ P_{k+1} = \bar{P}_{k+1} - KSK^{\intercal} \end{cases} \tag{5}$$

- Where
  - $\mathbf{z}_{k+1} - h\left(\bar{\mathbf{x}}_{k+1}\right)$ is called the **innovation**
  - S is the innovation covariance given by:
    - $S = J_h\left(\bar{\mathbf{x}}_{k+1}\right)\bar{P}_{k+1}J_h^{\intercal}\left(\bar{\mathbf{x}}_{k+1}\right) + R$
  - K is the Kalman gain given by:
    - $K = \bar{P}_{k+1}J_h^{\intercal}\left(\bar{\mathbf{x}}_{k+1}\right)S^{-1}$
  - Where $J_h\left(\bar{\mathbf{x}}_{k+1}\right)$
    - is the Jacobian of function h with respect to $\mathbf{x}$ evaluated at $\bar{\mathbf{x}}_{k+1}$.

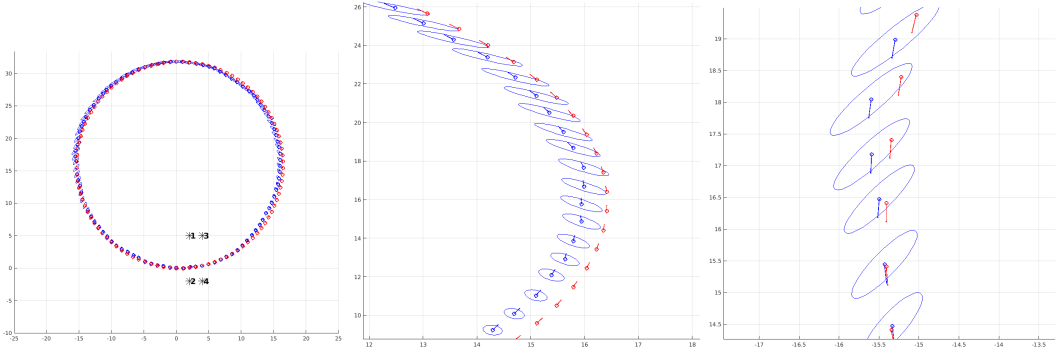
Example of EKF application

- 4 beacons with two of them made as always visible
- Observation model uses both range and bearing
- 5 robot poses (and moments of input control)
- Motion (process) model uses a simple differential drive robot
- Uncertainties were made all equal to 0.1 (large, mainly for angular entities)
- Uncertainty ellipses show the limits of the covariance of the localization



Another example of EKF application

- Same conditions as earlier...
- ...but using 100 poses along a uniform circular motion!
- Uncertainty ellipses now show larger deviations in some points
- In some parts landmarks are very far away and accuracy degrades!



Limitations of EKF in Occupancy Grid Maps

- Two situations where EKF is unsuited for robot localization:
  - When the environment is represented by an occupancy grid.
    - Sensor model for occupancy grid maps is not an analytic model (based on numerical process of ray casting), thus unsuitable for EKF.
    - There are however works (like Lakshitha Dantanarayana et al. 2015) with an alternative sensor model usable in occupancy grids.
  - When initial robot location is completely unknown.
    - Here, the location of the robot needs an arbitrary probability distribution; thus, the Gaussian assumption is violated.
- One possible strategy is to discretize the space of possible robot locations and thus deal with discrete probability distribution.
  - This method is known as Markov localization.
  - The computation burden associated with Markov localization is proportional to the size of the environment and the resolution of the discretization, making this strategy unsuitable in many situations.

Particle Filters

Particle Filters (Monte Carlo) Localization

- In Particle Filter Localization a weighted set of robot location estimates, named **particles**, is used to describe the probability distribution of robot location.
- Particle filters provide a more efficient alternative to Markov localization.
- The number of particles determines the accuracy of the representation.
- However, increasing the number of particles to obtain a higher accuracy leads to a more costly estimation process!
- Each particle provides a guess to the location of the robot.
- So, each particle is represented by three variables (x, y, θ) for a robot operating in a two-dimensional plane.

Principle of Particle Filters - 1

- Each particle i has a weight w_i
  - w_i indicates the contribution of particle i to the probability distribution.
- The sum of the weights of all particles is set to 1:
  - $\sum_{i=1}^{N} w_i = 1$ (N is the number of particles)
- A collection of such guesses describes the best knowledge available of the robot true location.
  - This is usually termed the **belief**,

Principle of Particle Filters - 2

- In the case of global localization, the initial robot location is completely unknown;
- Therefore, all locations of the environment are equally likely to contain the robot.
- Thus, initially, a set of equally weighted particles uniformly distributed in the environment is used to represent the belief of the robot location.
- During localization process, this belief is updated as more information is acquired from the sensors.
- Every time information from the sensors is gathered, the current belief is updated.
- The Particle Filter process has three steps:
  - Prediction
  - Update
  - Resampling

Particle Filter – Prediction

- When the robot is commanded to move:
  - a new belief is obtained by moving each particle
  - using the motion model equation (eq. (2))
  - δV_k and $\delta\omega_k$ are randomly generated.

Particle Filter – Update

- New sensor observation $\rightarrow$ update belief using an observation model.
- Particles weights are changed to reflect the likelihood L_i that the true robot location coincides with the corresponding particle.
- For j^{th} observation from a laser range finder, ray casting from each particle is used to obtain an expected measurement $\hat{d}_j$.
- With measurement d_j and sensor noise with zero mean and variance σ_d^2, the likelihood L_i can be computed using a Gaussian distribution based on:
  - $\frac{1}{\sigma_d \sqrt{2\pi}} \exp\left[-\frac{\left(\hat{d}_j - d_j\right)^2}{2\sigma_d^2}\right]$
- Likelihood of obtaining a sequence of observations is computed by multiplying together all the likelihoods.
- Once likelihoods of all the particles are computed, these are normalized to obtain the weight of each of the particles.
  - $w_i = \frac{w_i L_i}{\sum\limits_{j=1}^{N} w_j L_j}$

Particle Filter – Resampling

- Performed to avoid the situation where a small number of particles with large weights dominate the representation of the belief.
- One common strategy used for resampling is as follows:
  - Compute an estimate of the effective number of particles as
    - $n_{\text{eff}} = \frac{1}{\sum_{i=1}^{n} w_i^2}$
  - If n_{eff} is less than a threshold, then draw n particles from the current particle set with probabilities proportional to their weights.
  - Replace the current particle set with this new one.
  - Set the weights of each particle to be $1/n$.
- The resulting set of particles represents the updated belief of the robot location.
  - Use either the particle with largest weight or
  - Combine all particles to estimate robot pose (weighed sum of particles)
    - $\hat{\mathbf{x}} = \sum_i w_i \mathbf{x}_i^p$, being $\mathbf{x}_i^p$ the pose variables of particle i.

Introduction to Markov Localization

Markov Localization - basics

- In Markov localization the state space is discretized into a grid and the probability that the robot is present in a particular grid cell is used to describe the estimate of the robot location.

- In a 2D scenario, the discretization is over three–dimensional space incorporating the robot position and orientation.
- At time k, the probability that the robot is present in each of the grid cells is represented by
  - $p_i\left(k\right) \triangleq P\left(\mathbf{x}_k = i\right)$, $i = 1, \cdots, M$
- where
  - $P(\mathbf{x}_k = i)$ means the probability of robot pose $\mathbf{x}_k$ is in grid cell i,
  - M is the total number of grid cells, and
  - $0 \leqslant p_i\left(k\right) \leqslant 1$, $\sum\limits_{i=1}^{M} p_i\left(k\right) = 1$
- This probability distribution is called belief $\mathrm{bel}(k)$.

Initializing Markov Localization

- Initial belief $\mathrm{bel}(0)$ is the prior distribution.
- When there is no prior knowledge about the robot location, the probability distribution is a uniform one. That is,
  - $p_i(0) = \frac{1}{M},\ i = 1, \cdots, M$
- Given:
  - a belief $\mathrm{bel}(k)$ at time k
  - and a new control input $\mathbf{u}_k$
  - and a new observation $\mathbf{z}_{k+1}$
- the belief needs to be updated to find $\mathrm{bel}(k+1)$ using Bayes filter.
- There are two essential steps used to update the belief:
  - Prediction
  - Update

Prediction of Markov belief

- The new control input $\mathbf{u}_k$ and the previous belief $\mathrm{bel}(k)$ are used to compute the predicted belief $\mathrm{bel}\left(k+1\right)$ that can be computed with:
  - $\overline{p_j\left(k+1\right)} = \sum_{i=1}^{M} p_i\left(k\right) P\left(\mathbf{x}_{k+1} = j | \mathbf{x}_k = i, \mathbf{u}_k\right)$
- This equation is obtained using the law of total probability.
- Here $P\left(\mathbf{x}_{k+1} = j | \mathbf{x}_k = i, \mathbf{u}_k\right)$ is the conditional probability that can be obtained from the motion model.

Update of Markov belief

- Using Bayes' theorem, information in the new observation $\mathbf{z}_{k+1}$ is fused with the prediction to obtain the new belief at time k+1 as follows:
  - $p_j\left(k+1\right) \triangleq P\left(\mathbf{x}_{k+1}=j|\mathbf{z}_{k+1}\right) = \frac{P(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}=j)\overline{p_j(k+1)}}{\sum\limits_{i=1}^{M} P(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}=i)\overline{p_i(k+1)}}$
- where the conditional probability $P\left(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}=i\right)$ can be obtained from the sensor model.