Probabilistic Fundamentals in Robotics

Probabilistic Models of Mobile Robots
Robotic mapping

Basilio Bona
DAUIN – Politecnico di Torino
July 2010
Course Outline

- Basic mathematical framework
- Probabilistic models of mobile robots
- Mobile robot localization problem
- Robotic mapping
- Probabilistic planning and control

Reference textbook
http://www.probabilistic-robotics.org/
Robotic mapping

- Occupancy grid mapping
- SLAM
  - GraphSLAM
  - Sparse extended information filter
  - FastSLAM
In many cases maps are already available and are used to perform robot localization: we have already discussed the localization problem with known maps.

However in many cases *maps are unknown* and must be built by the robot itself as it moves in the environment.

Since odometry is inexact, the true pose of the robot is not known with precision, therefore mapping is and localization must be performed at the same time, post-processing the robot measurements.

The *concurrent process* of localization and mapping is called SLAM (simultaneous localization and mapping).
Example

- The robot has to acquire the map of the environment while localizing itself relative to this map.
- What happens if we simply perform localization using the filters learned so far and we do the mapping with the (uncertain) pose estimate?

![Raw range data, position indexed by odometry](image)

![Occupancy grid map](image)

post-process
Map types

Landmarks

Digital elevation map

Landmark-based representation (stochastic map)

Occupancy grid-map

Point-cloud representation
Two particular representations are common in the applications of mobile robotics in indoor scenarios:

- **Landmark-based maps**: stochastic maps that contain a probabilistic description (usually mean+covariance) of the position of salient features of the scenario.
- **Occupancy grid maps**: high-resolution models of the environment; it is a grid in which each cell contains the probability of the cell being occupied.
3D mapping

Other types of maps, as 3D mapping are possible, but computationally much more complex.
Assuming that we know the robot poses, mapping is the task of building a consistent representation of the environment.

Depending on the scenario it is convenient to use different world representations.

*Occupancy grid maps* are the best choice for maps, and the related family of algorithms is called occupancy grid mapping.
The simplest case is introduced first: *the robot pose is known* but the measurements are noisy.

2D maps are the most common; they can be used when robot motion takes place on a flat surface and sensors capture essentially a slice of the surrounding environment.

**Occupancy grid maps**
In principle, the algorithm shall compute the posterior probability of the map

\[ p(m \mid z_{1:t}, x_{1:t}) \]

The (occupancy grid) map is described by a set of \( N \) cells

\[ m = \{ m_1, m_2, \ldots, m_N \} \]

At each cell a probability of occupancy (0 = free - white, 1 = occupied - black) is attached

\[ p(m_i) \]

The mapping problem is decomposed into \( N \) simpler problems

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

This decomposition does not allow to represent dependencies among neighboring cells, and we throw away much information.
The previous factorization transforms the problem into a binary estimation problem with static state; a binary Bayes filter is adopted.

The log odds representation of occupancy is used; the advantage is that numerical instabilities near 0 or 1 are avoided.

\[
l_{t,i} = \log \frac{p(m_i | z_{1:t}, x_{1:t})}{1 - p(m_i | z_{1:t}, x_{1:t})}
\]

\[
p(m_i | z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp(l_{t,i})}
\]
Occupancy grid mapping

\[ \text{Occupancy}_\text{grid}_\text{mapping}(\{l_{t-1,i}\}, x_t, z_t) \]

1:  forall cells \( m_i \)

2:  if \( m_i \) in sensor field of \( z_t \) then

3:  \[ l_{t,i} = l_{t-1,i} + \log \frac{p(m_i | z_t, x_t)}{1 - p(m_i | z_t, x_t)} - l_0 \]

4:  else

5:  \[ l_{t,i} = l_{t-1,i} \]

6:  endif

7:  endfor

return \( \{l_{t,i}\} \)

This is the inverse sensor model in log form.
Inverse sensor model

`Inverse_sensor_model(m_i, x_t, z_t)`

1: Let $\bar{x}_i, \bar{y}_i$ be the c.o.m. of $m_i$

2: $r = \sqrt{(\bar{x}_i - x)^2 + (\bar{y}_i - y)^2}$

3: $\phi = \text{atan2}(\bar{y}_i - y, \bar{x}_i - x) - \theta$

4: $k = \arg \min_j |\phi - \theta_{j,\text{sens}}|$

5: if $r > \min(z_{max}, z^k_t + \alpha)$ or $|\phi - \theta_{j,\text{sens}}| > \beta/2$ then

6: return $l_0$ outside range

7: if $z^k_t < z_{max}$ and $|r - z^k_t| < \alpha/2$

8: return $l_{\text{occ}}$ occupied cells

9: if $r \leq z^k_t$

10: return $l_{\text{free}}$ free cells

11: endif

$x_t = (x, y, \theta)$

robot pose

Basilio Bona  
July 2010
Parameters of the inverse sensor model

range closed to the measured range

sensor cone

k-th beam

\( \bar{x}, \bar{y} \)

\( x, y \)
Example using ultra-sound sensors

Initial map

Final map
Another example
Multi-sensor fusion

- When robots have different types of onboard sensors, it is natural to try to integrate their measurements.
- Sensors may have different characteristics, i.e., different models of the perceived world; sometimes an obstacle detected by one sensor is not detected by a different sensor.
- To avoid the occurrence of conflicting information, a popular solution builds separate maps for each sensor type and then integrates them using a suitable combination function.
Multi-sensor fusion

Let the map built by the $k$-th sensor type be

$$m^k = \{m^k_1, m^k_2, \ldots, m^k_N\} = \{m^k_i\}$$

- If the sensors measurements are independent, we can combine them using the

  *De Morgan’s Law*

  \[
  \text{NOT (P OR Q)} = (\text{NOT P}) \text{ AND (NOT Q)}
  \]
  \[
  \text{NOT (P AND Q)} = (\text{NOT P}) \text{ OR (NOT Q)}
  \]

  \[
  p(m_i) = 1 - \prod_k (1 - p(m^k_i))
  \]

- Alternatively one can compute the most *pessimistic* estimate coming from the sensors

  \[
  p(m_i) = \max_k p(m^k_i)
  \]
Simultaneous localization and mapping (SLAM)

- SLAM is also known as *Concurrent Mapping and Localization* or *CML*.
- The problem arises when the robot does not have access to the environment map and does not know its pose.
- So, both the map and the pose must be estimated concurrently from measurements $z_{1:t}$ and controls $u_{1:t}$.
- SLAM is a very complex problem, significantly harder than simple localization or simple mapping with known pose.
- Many types and algorithms have been studied; some are incremental, some are complete.
We need to estimate the joint distribution of both robot pose and map representation of the environment:

\[ \text{bel}_t = p(x_t, m \mid z_{1:t}, u_{1:t}) \]

This is called **online** SLAM problem since it discards past measurements.

The **full** SLAM problem considers the entire path:

\[ \text{bel}_t = p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \]
Online SLAM

Estimated variables
Full SLAM

\[ u_{t-1} \rightarrow x_{t-1} \rightarrow x_t \rightarrow x_{t+1} \]

\[ u_t \]

\[ z_{t-1} \]

\[ z_t \]

\[ z_{t+1} \]

Estimated variables

\[ m \]
The relation between the two approaches is simple to state

\[
p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \cdots dx_{t-1}
\]

Moreover, its nature is both continuous and discrete

- The objects on the map and the robot pose are continuous
- The correspondence between a measurement and a feature in the map or between a feature and a previously detected one is discrete

If the correspondence must be made explicit, we write

\[
p(x_t, m, c_t \mid z_{1:t}, u_{1:t}) \quad \text{or} \quad p(x_{1:t}, m, c_t \mid z_{1:t}, u_{1:t})
\]
Problems arise from
- High dimensionality of the continuous parameter space
- Large number of discrete correspondence variables

The algorithms employed depend mainly on which type of map needs to be estimated
- Landmark-based maps
  - EKF-SLAM
  - Sparse Extended Information Filters
  - GraphSLAM
  - Rao-Blackwellized Particle Filters (FastSLAM)
- Occupancy grid maps
  - Rao-Blackwellized Particle Filters
  - GraphSLAM
EKF SLAM with landmark-based maps

- Assumptions:
  - Feature (i.e., landmark) based maps; the number of landmarks must be relatively small (< 1,000)
  - Works well if landmarks are unambiguous; feature detectors must be optimized
  - It is based on Gaussian noise assumption for robot motion and perception
  - Makes linearization, so the amount of uncertainty of the posterior shall be limited
  - Can only process *positive* sightings of landmarks (no information from absence of landmarks)
  - Landmark correspondences are exactly known
EKF SLAM with landmark-based maps

- EKF SLAM includes the position of the landmarks in the state vector and performs estimation over the **augmented state**:

\[
\mathbf{y}_t = \begin{pmatrix} \mathbf{x}_t \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} x & y & \theta \\ \ell_{1,x} & \ell_{1,y} & \ell_{2,x} & \ell_{2,y} & \cdots & \ell_{N,x} & \ell_{N,y} \end{pmatrix}
\]

- Observing a particular landmark improves the position estimate of all landmarks, not only this one

- This improvement is mediated by the robot pose: landmark observation improves the robot pose, that in turn improves the localization of previous observed landmarks
Mathematically, this dependence is captured by the off diagonal elements of the covariance matrix.

\[ x_t \sim N \begin{pmatrix} 
\mu_{x_r} \\
\mu_{y_r} \\
\mu_{\theta_r} \\
\mu_{l_{1x}} \\
\mu_{l_{1y}} \\
\vdots \\
\mu_{l_{Nx}} \\
\mu_{l_{Ny}} 
\end{pmatrix}, \quad 
\begin{pmatrix} 
\sigma_{x_r x_r} & \sigma_{x_r y_r} & \sigma_{x_r \theta_r} & \sigma_{x_r l_{1x}} & \sigma_{x_r l_{1y}} & \cdots & \sigma_{x_r l_{Nx}} & \sigma_{x_r l_{Ny}} \\
\sigma_{x_r y_r} & \sigma_{y_r y_r} & \sigma_{y_r \theta_r} & \sigma_{y_r l_{1x}} & \sigma_{y_r l_{1y}} & \cdots & \sigma_{y_r l_{Nx}} & \sigma_{y_r l_{Ny}} \\
\sigma_{x_r \theta_r} & \sigma_{y_r \theta_r} & \sigma_{\theta_r \theta_r} & \sigma_{\theta_r l_{1x}} & \sigma_{\theta_r l_{1y}} & \cdots & \sigma_{\theta_r l_{Nx}} & \sigma_{\theta_r l_{Ny}} \\
\sigma_{x_r l_{1x}} & \sigma_{y_r l_{1x}} & \sigma_{\theta_r l_{1x}} & \sigma_{l_{1x} l_{1x}} & \sigma_{l_{1x} l_{1y}} & \cdots & \sigma_{l_{1x} l_{Nx}} & \sigma_{l_{1x} l_{Ny}} \\
\sigma_{x_r l_{1y}} & \sigma_{y_r l_{1y}} & \sigma_{\theta_r l_{1y}} & \sigma_{l_{1y} l_{1x}} & \sigma_{l_{1y} l_{1y}} & \cdots & \sigma_{l_{1y} l_{Nx}} & \sigma_{l_{1y} l_{Ny}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\sigma_{x_r l_{Nx}} & \sigma_{y_r l_{Nx}} & \sigma_{\theta_r l_{Nx}} & \sigma_{l_{1x} l_{Nx}} & \sigma_{l_{1y} l_{Nx}} & \cdots & \sigma_{l_{Nx} l_{Nx}} & \sigma_{l_{Ny} l_{Nx}} \\
\sigma_{x_r l_{Ny}} & \sigma_{y_r l_{Ny}} & \sigma_{\theta_r l_{Ny}} & \sigma_{l_{1x} l_{Ny}} & \sigma_{l_{1y} l_{Ny}} & \cdots & \sigma_{l_{Nx} l_{Ny}} & \sigma_{l_{Ny} l_{Ny}} 
\end{pmatrix} \]
We can simply extend the expression of the dynamic system used for EKF Localization:

\[
\begin{align*}
\mathbf{p}_t &= f(\mathbf{p}_{t-1}, \mathbf{u}_t, \mathbf{\omega}_t) \\
\mathbf{z}_t &= h(\mathbf{p}_t, \text{map}, \mathbf{v}_t)
\end{align*}
\]

So the estimation procedure can proceed according to EKF, iterating prediction and update phase.
EKF-SLAM: a typical situation

(1) Robot starts; first measurement of feature A

(2) Robot drives forward (uncertainty grows)

(3) Robot makes first measurements of B and C

State vector $x_t$ can grow as new landmarks are discovered

(4) Robot drives back towards the start (uncertainty grows)

(5) Robot re-observes A (loop closure); uncertainty shrinks

(6) Robot re-observes B; also the uncertainty of C shrinks
Another example: EKF applied to the online SLAM problem

The robot’s path is a dotted line, and its estimates of its own position are shaded ellipses. Eight distinguishable landmarks of unknown location are shown as small dots, and their location estimates are shown as white ellipses.

In (a)–(c) the robot’s positional uncertainty is increasing, as is its uncertainty about the landmarks it encounters.

In (d) the robot senses the first landmark again, and the uncertainty of all landmarks decreases, as does the uncertainty of its current pose.
When landmark correspondence is unknown, the EKF SLAM algorithm is extended using an incremental maximum likelihood estimator to determine landmark correspondence.

A new landmark is created if the Mahalanobis distance to all existing landmarks in the map exceeds a value $\alpha$.

Outliers detection: how to detect spurious landmarks that are outside the uncertainty range of landmarks already present.

EKF is fragile with respect to landmark confusion, so one of the following methods is used to avoid it:

- Spatial arrangement: choose landmarks that are far apart so that it is unlikely to confuse them -> optimal tradeoff between to few and too many landmarks.
- Signatures: chose landmarks with distinguishable signatures (colors, dimensions, etc.)
EKF: summary

- EKF landmark-based SLAM is an effective and easy to implement solution to many problems
- It has been successfully applied also in large-scale environments
- Online version ties to determine the momentary robot pose
- Global problem ties to determine all poses
- With known correspondence the algorithm is incremental
- Maps management wrt outliers
  - Include a provisional of landmarks that are not observed sufficiently often
  - Include a landmark evidence counter that computes the posterior evidence of the existence of a landmark
EKF: summary

- **Critical Issues**
  - Complexity is quadratic in the number of landmarks: $O(n^2)$
  - Can diverge if nonlinearities are large
  - Data association: how do we decide if we are observing an already seen landmark?
Rao-Blackwellized Particle Filters (RBPF-SLAM)

- Rao-Blackwellized Particle Filters (RBPF), also known as FastSLAM, are a sample-based techniques for solving SLAM.
- They are particularly suitable for estimating occupancy grid-maps of an indoor environment.
- Rao-Blackwellized Particle Filters are based on the following factorization of the SLAM belief:

\[
bel_t = \text{prob}(x_{1:t}, m | z_{1:t}, u_{1:t}) = \\
\text{prob}(m | x_{1:t}, z_{1:t}) \cdot \text{prob}(x_{1:t} | z_{1:t}, u_{1:t})
\]
Rao-Blackwellized Particle Filters (RBPF-SLAM)

- The previous formula is known as Rao-Blackwell factorization, which gives the name to the corresponding solution to SLAM.
- The underlying math translates in the following simple concept:

  **RBPF estimates potential trajectories of the robot and for each hypothesis the SLAM reduces to mapping with known poses.**

The filter estimates potential trajectories and a map is associated to each trajectory hypothesis.
Rao-Blackwellized Particle Filters (RBPF) have been demonstrated to be an effective solution for the estimation of occupancy grid maps.

No data association (grid maps does not distinguish elements in the environment)

**Issues**

- Each particle carries on a complete map hypothesis: a normal computer is not able to manage more than few hundred particles.
- But to cope with large uncertainty you need a large number of particles.
Thank you.
Any question?