Complexity Issues in Probabilistic Mapping

João P. Hespanha
Center for Control Engineering and Computation
University of California at Santa Barbara, USA

Motivating problems

Risk avoidance:  
How to go from $A$ to $B$ avoiding the $!!!$?

Search:  
How to find the $!!!$ as fast as possible?

Decisions are made based on noisy data collected by the $!!!$ regarding the position of the $!!!$. Sensors are not perfect.

Uncertainty in map of objects’ positions constructed from sensor data.
Probabilistic maps

**probabilistic map**: conditional distribution of objects’ positions given the available measurements

\[ m_r(r | Y) := P(r(t_2) = r | Y(t_1) = Y) \]

objects’ position at time \( t_2 \)

measurements collected up to time \( t_1 \)

Outline

- Mathematical framework
- Sensor model
- Mapping static objects
- Mapping dynamic objects
- Application to minimum-risk path planning
- Application to optimal search
Computing probabilistic maps

\( \mathcal{R} \) \( \equiv \) region being mapped (partitioned into cells)
\( r := \{ r_1, r_2, \ldots, r_n \} \) \( \equiv \) positions of \( n \) objects of the same type. Each \( r_i \in \mathcal{R} \)
\( y_1, y_2, y_3, \ldots \) \( \equiv \) sequence of measurements taken by sensors

Goal: Given a list of measurements \( Y_k := \{ y_1, y_2, \ldots, y_k \} \) collected up to time \( t_k \), compute the a posteriori conditional distribution of \( r \):

\[
m_r( \mathbf{r} | Y_k ) := P( \mathbf{r}(t) = \mathbf{r} | Y_k ), \quad t \geq t_k
\]

Sensor model

\( y_1, y_2, y_3, \ldots \) \( \equiv \) sequence of measurements taken by sensors.
Each \( y_k = \{ t_k, a_k, s_k \} \)

\( a_k = 0 \) \( \equiv \) no object seen
\( s_k \in \mathcal{R} \) \( \equiv \) object detected at position \( s_k \)

Sensor parameters (not necessarily constant over time or space):

\( \rho \) sensor range
\( p_1 \) probability of not recognizing an object within range (false negative)
\( p_2 \) probability of recognizing a “distractor” instead of a real object when an object is within sensor range (false positive)
\( p_3 \) probability of recognizing a “distractor” instead of a real object when no object is within sensor range (false positive)
\( e \) maximum localization error, when the sensor detects a real object (typically \( e \ll \rho \))

objects probabilistic maps

Sensor model

objects are indistinguishable (\( s_k \in \mathcal{R} \) provides no information regarding which object was seen)
Sensor model

\( y_1, y_2, y_3, \ldots \equiv \) sequence of measurements taken by sensors.
Each \( y_k = \{ t_k, a_k, s_k \} \)

\( n \)-object sensor likelihood function

\[ \ell_n(y, r) := P(y_k = y \mid r(t) = r) \quad y := \{ t, a, s \} \]

each measurement \( y_k \) is conditionally independent of any other measurement given an object configuration \( r(t_k) \)

Sensor model (single object)

\( y_1, y_2, y_3, \ldots \equiv \) sequence of measurements taken by sensors.
Each \( y_k = \{ t_k, a_k, s_k \} \)

\( n \)-object sensor likelihood function

\[ \ell_n(y, r) := P(y_k = y \mid r(t) = r) \quad y := \{ t, a, s \} \]

For the single object case \((n = 1)\):

\[ \ell_1(y, r) = \begin{cases} \frac{1-p_1 \cdot p_2}{B} & |r - a| \leq \rho, \ |s - r| \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \]

\[ \ell_1^2 \leq \frac{E_x \cdot E_y}{A^2} \]

\[ |r - a| \leq \rho, \ |s - a| \leq \rho \]

\[ |r - a| > \rho, \ |s - a| \leq \rho \]

\[ |r - a| \leq \rho, \ |s - a| \leq \rho \]

\[ 0 \]

otherwise

Sensor parameters

\( \rho \) sensor range
\( p_1 \) prob. not recognizing an object within range
\( p_2 \) prob. recognizing “distractor” instead of real object when an object is within range
\( p_3 \) prob. recognizing “distractor” instead of real object when no object is within range
\( e \) max. localization error, when the sensor detects a real object (typically \( e \ll \rho \))
Typical single-object sensor likelihood function

\[ \ell_1(t, a, s, r) \]

\[ \ell_2(t, a, s, r) \]

\[ \ell_3(t, a, s, r) \]

very much non-Gaussian...

Sensor model (multiple objects)

\[ \ell_n(y, r) := P(y_k = y | r(t) = r) \quad y := \{t, a, s\} \]

Sparseness assumption

1. minimum distance between any two objects \( d_{\text{min}} \geq 2 \rho \)
   (no two objects in sensor range)

or 1’ global sensor (\( \rho = +\infty \)) and \( d_{\text{min}} \geq 2 e \)
   (no two objects within area of localization uncertainty)

or 1'' sensor only produces positive readings and \( d_{\text{min}} \geq 2 e \)

Theorem: For every \( r \in \mathcal{R}^n \) for which no two objects are closer then \( d_{\text{min}} \)

normalizing constant

\[ \ell_n(y_k, r) = \epsilon \prod_{i=1}^n \ell_1(y_k, r_i), \quad \forall k \]

single measurement \( \{t_k, a_k, s_k\} \)

possible configuration for the \( n \) objects \( r := \{r_1, \ldots, r_n\} \)

single-target sensor likelihood function
Mapping static objects

When \( r(t) = r(0) \) \( \forall t \geq 0 \ldots \)

\[
m_r(r | Y) := P(r = r | Y_k = Y) \quad r := \{ r_1, r_2, \ldots, r_n \}
\]

\[
e^{-\sum_{j=1}^{n} f(r_j | Y)} p_{0}(r)
\]

\[
f(x | Y_k) := \prod_j \ell_j(y_j, x), x \in \mathcal{R}
\]

1. \( m_r(\cdot | Y) \) is defined on \( \mathcal{R}^n \) thus the memory complexity to represent it is \( o(N^n) \)
2. \( f(\cdot | Y) \) is defined on \( \mathcal{R} \) thus the memory complexity to represent it is only \( o(N) \)
3. \( f(\cdot | Y) \) can be computed efficiently recursively (one measurement at a time)

\[
f(x, \{ y_1, y_2, \ldots, y_{k+1} \}) = \ell_{k+1}(y_{k+1}, x) f(x, \{ y_1, y_2, \ldots, y_k \})
\]

Moving objects

Markovian motion model

\[
\Phi(r', r; t + dt, t) := P(r(t + dt) = r' | r(t) = r)
\]

\( n \)-object transition probability function

Can take into account:
1. uncertainty in the motion
2. regions of high/low mobility (e.g., roads versus, rocky terrain)
3. preferential directions of motion

The \( n \)-object transition probability function \( \Phi \) is such that no two objects are closer than \( d_{\min} \) with probability one for all \( t \geq t_0 \).
Recursive map building

The prediction operator takes a map at time $\tau$ and propagates it forward in time to obtain a map at time $t > \tau$

$$m_r(\cdot; t\| Y) = T_{\text{pred}}[t, \tau, m_r(\cdot; \tau\| Y)]$$

The fusion operator takes a map at time $t_i$ and, given a list of measurements $Y_{i-1}$ taken before $t_i$ and a new measurement $y_i$ taken at $t_i$, produces a new map given all measurements

$$m_r(\cdot; t_i\| Y_i) = T_{\text{fuse}}[y_i, m_r(\cdot; t_i\| Y_{i-1})]$$

Map building operators

The fusion operator is defined by:

$$m_r(\cdot; t_i\| Y_i) = c \ell_n(y_i, \cdot) m_r(\cdot; t_i\| Y_{i-1})$$

The prediction operator is defined by:

$$m_r(\cdot; t + dt\| Y_i) = \sum_{r' \in R^n} \Phi(r', r; t + dt, t) m_r(\cdot; t\| Y_i)$$

$R$ - set of $n$-dimensional objects

$\ell_n$ - object sensor likelihood function (sensor model)

$\Phi$ - object transition probability function (motion model)

$c$ - normalizing constant

$\Phi(r', r; t + dt, t)$ - transition probability of an object from $r$ to $r'$ in time $dt$

$\sum_{r' \in R^n}$ - sum over all $n$-dimensional objects

To use these formulas directly one needs an extensive representation of the map – memory and computational complexity $O(N^n)$

$N$ - number of objects

$R$ - set of $n$-dimensional objects

$O(N^n)$ - computational complexity
Separable maps

**Definition:** A probabilistic map is called separable if it can be written as

\[
m_r(r; t|Y) := c \left( \prod_{j=1}^{n} f(r_j; t|Y) \right) \delta(r; d_{\text{min}})
\]

\[
\delta(r; d_{\text{min}}) := \begin{cases} 
0 & \exists i, j : |r_i - r_j| < d_{\text{min}} \\
1 & \text{otherwise}
\end{cases}
\]

From previous result: for stationary objects and the class of sensors considered here, the probabilistic maps are separable

*still approximately true for mobile objects*

---

**Fusion operator for separable maps**

Assume map at time \(t_i\), given measurements \(Y_{i-1}\) is separable

\[
m_r(r; t_i|Y_{i-1}) := c \left( \prod_{j=1}^{n} f(r_j; t_i|Y_{i-1}) \right) \delta(r; d_{\text{min}})
\]

map at time \(t_i\), given measurements \(Y_{i-1}\)

**Theorem:** The fusion operator preserves separability, i.e.,

\[
m_r(r; t_i|Y_i) := c \left( \prod_{j=1}^{n} f(r_j; t_i|Y_i) \right) \delta(r; d_{\text{min}})
\]

map at time \(t_i\), given measurements \(Y_i\)

where

\[
f(r; t_i|Y_i) := t_{\text{S}}(y_i, r) f(r; t_i|Y_{i-1})
\]

\[
\{y_i, Y_{i-1}\} \text{ new measurement at } t_i
\]

\[
\{r, Y_i, Y_{i-1}\} \text{ measurements before/at } t_i
\]

*fusion can be efficiently done for large number of objects*
**Prediction operator for separable maps**

- **Bounded velocity assumption**
  - \( \phi_1(r'; r; t + dt, t) := \begin{cases} 1 - o(dt) & r = r' \\ o(dt) & r \neq r' \\ dt \to 0 \end{cases} \)

- **Independent motion assumption**
  - \( \Phi(r'; r; t + dt, t) = c(r) \left( \prod_{k=1}^{n} \phi_1(r'_k, r_k; t + dt, t) \right) \delta(r'; d_{min}) \)

**Theorem:** The prediction operator approximately preserves separability, i.e.,

\[
m_r(r; t + dt | Y_i) = c \left( \prod_{j=1}^{n} f(r_j; t + dt | Y_i) \right) \delta(r; d_{min}) + \epsilon(r; dt) \quad \forall r \in \mathbb{R}^n
\]

where

- \( \epsilon(r; dt) = o(dt^2) \) except for configuration with all objects further apart than \( d_{min} \) but for which this could be violated by the motion of a single object (in which case \( \epsilon(r; dt) = o(dt) \))

**Prediction can be efficiently done for large number of objects**
Summary so far…

Probabilistic mapping can be done efficiently for large number of objects with memory and computational complexity $o(N)$ [and not $o(N^n)$]

1. Under some form of **sparseness assumption**, which imposes a minimum distance between any two objects
2. Assuming a **Markovian motion** model, under which objects essentially move independently while they are away from each other
3. The map will exhibit an **error** essentially $o(dt^2)$

Application to minimum-risk path planning

Robot navigates over the region $\mathcal{R}$ populated by objects

Each object induces danger if robot comes close to it

**Risk parameters** (not necessarily constant over time or space):
- $\rho_{\text{danger}}$ range over which object is dangerous
- $P_{\text{destroy}}$ probability that robot will be destroyed if it comes close to object

**Goal**: compute path from initial position $A$ to final position $B$ that maximizes probability of $P_{\text{survive}}$ of not being damaged
Application to minimum-risk path planning

Risk parameters (not necessarily constant over time or space):
- $\rho_{\text{danger}}$: range over which object is dangerous
- $p_{\text{destroy}}$: Probability that robot will be destroyed if it comes close to object

Goal: compute path from initial position $A$ to final position $B$ that maximizes probability of $p_{\text{survive}}$ of not being damaged

Given a path: $x_1 = A, x_2, x_3, \ldots, x_m = B \in \mathcal{R}$

\[
p_{\text{survive}} = \prod_{j=1}^{m} (1 - m_{\text{danger}}(x_j | Y))
\]

\[
m_{\text{danger}}(x | Y) := P(\text{robot destroyed when passing through } x | Y_k = Y, x \in \mathcal{R})
\]

Minimum-risk path planning

Given a path: $x_1 = A, x_2, x_3, \ldots, x_m = B \in \mathcal{R}$

\[
p_{\text{survive}} = \prod_{j=1}^{m} (1 - m_{\text{danger}}(x_j | Y)) \quad \Rightarrow \quad \log p_{\text{survive}} = \sum_{j=1}^{m} \log_{\text{survive}}(x_j | Y)
\]

Minimum risk path

\[
\max_{x_1, x_2, \ldots, x_m: \quad x_1 = A, \ x_m = B, j = 1} \sum_{j=1}^{m} \log_{\text{survive}}(x_j | Y)
\]

Minimum risk path, subject to length constraint: $m \leq M$

Minimum length path, subject to risk constraint: $p_{\text{survive}} \geq p^*$

optimizations in graphs with additive cost/constraint
Computing danger maps

Danger map:

\[ m_{\text{danger}}(x|Y) := P(\text{robot destroyed when passing through } x \mid Y_k = Y) \]

\[ = \sum_{||z-x|| \leq \rho_{\text{danger}}} p_{\text{destroy}} m_{\rho}(z, Y) \]

Risk parameters (not necessarily constant over time or space):

- \( \rho_{\text{danger}} \): range over which object is dangerous
- \( p_{\text{destroy}} \): probability that robot will be destroyed if it comes close to object

Theorem:

\[ m_o(x, Y) \approx c f(x, Y) g_0(x) \left( n f_0(Y) \right)^{n-1} + \]

\[ \sum_{k=1}^{n} \frac{n! f_0(Y)^{n-k-1}}{(n-1-k)!} \sum_{\sigma \in S(k)[x]} f_{\sigma_1}(Y) f_{\sigma_2}(Y) \ldots f_{\sigma_k}(Y) \]

where \( f_i(Y) := \sum_{x \in R_i} f(x, Y) g_0(x) \)

Danger maps can be computed directly from the a.m.f. \( f(\cdot, Y) \)

Minimum-risk path planning

Goal: compute path from initial position \( A \) to final position \( B \) that maximizes probability of \( p_{\text{survive}} \) of not being damaged

From a complexity viewpoint, minimum risk path planning is a well-behaved problem...
Application to search

Robot navigates over the region $\mathcal{R}$ populated by objects

**Goal:** compute path starting in an initial position $A$ that maximizes probability of $p_{\text{find}}$ of finding an object in a given time interval.

---

Application to search

Robot navigates over the region $\mathcal{R}$ with a single static object

**Goal:** compute path starting in an initial position $A$ that maximizes probability of $p_{\text{find}}$ of finding the object in a given time interval.

Given a path: $x_1 = A, x_2, x_3, \ldots, x_m \in \mathcal{R}$

\[
p_{\text{find}} = \sum_{x \in \{x_1, x_2, \ldots, x_m\}} m_o(x, Y) \leq \sum_{i=1}^{m} m_o(x_i, Y)
\]

summed over path as a set (i.e., repeated cells are counted only once – seeing a position twice does not increase probability of finding the object)
Optimal search

Given a path: $x_1 = A, x_2, x_3, \ldots, x_m \in \mathcal{R}$

$$p_{\text{find}} = \sum_{x \in \{x_1, x_2, \ldots, x_m\}} m_o(x, Y)$$

probability that object will be found

Maximum probability path, subject to length constraint: $m \leq M$

$$\max_{x_1, x_2, \ldots, x_m; \sum x = A, m \leq M, x \in \{x_1, x_2, \ldots, x_m\}} \sum m_o(x, Y)$$

optimizations in graphs with non-additive cost/constraint

Minimum length path, subject to probability constraint: $p_{\text{find}} \geq p^*$

$$\min_{x_1, x_2, \ldots, x_m; \sum x = A, m} \sum_{x \in \{x_1, x_2, \ldots, x_m\}} m_o(x, Y) \geq p^*$$

approximating $P_{\text{find}}$ by the additive cost $\sum m_o(x_i, Y)$ will not work because it would lead the robot to the location with largest probability and keep it there…

A hierarchical algorithm...

Premise: If the map the distribution were uniform, sweeping would be optimal

Hierarchical algorithm:

1. Partition map into a (small) number of connected areas where the distribution is approximately uniform.

   one can use, e.g., state-aggregation algorithms for Markov chains

2. Solve the high-level optimal problem of deciding which areas to visit, in what order, and how long to stay in each one.

   still a combinatorial problem

3. Inside each area use a sweeping path (low-level)

   overall sub-optimal but computationally much better
Conclusions

Probabilistic maps can be economically represented and computed with complexity independent of the number of objects under “reasonable” assumptions (sparseness, Markovian motion)

Probabilistic maps can be used to efficiently solve minimum risk path planning problems

Optimal search problems are computationally much worse. Efficient sub-optimal hierarchical solutions are possible