2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

A Bayesian framework for optimal motion planning with uncertainty

Andrea Censi, Daniele Calisi, Alessandro De Luca, Giuseppe Oriolo

Abstract— **Modeling robot motion planning with uncertainty in a Bayesian framework leads to a computationally intractable stochastic control problem. We seek hypotheses that can justify a separate implementation of control, localization and planning. In the end, we reduce the stochastic control problem to path-planning in the extended space of poses $\times$ covariances; the transitions between states are modeled through the use of the Fisher information matrix. In this framework, we consider two problems: minimizing the execution time, and minimizing the final covariance, with an upper bound on the execution time. Two correct and complete algorithms are presented. The first is the direct extension of classical graph-search algorithms in the extended space. The second one is a back-projection algorithm: uncertainty constraints are propagated backward from the goal towards the start state.**

I. Introduction

This paper considers the problem of planning a safe path for a mobile robot, in spite of noisy odometry and sensor readings. The difficulty in approaching this problem is twofold. Firstly, with respect to the regular path-planning problem, there is considerable more machinery needed to formalize uncertainty: this implies some design decisions even at the modeling stage. Then, the resulting model is a stochastic control problem which is very hard to solve in the general case, hence many approximations are needed to turn it into an approachable problem.

This paper tries to solve the problem by working in the extended space of poses$\times$covariances, as has already been done in [1], [2]. The following are the main contributions of this paper with respect to previous work.

- We model the problem in a Bayesian framework and we formalize transitions between information states by using the Fisher information matrix. We look for the hypotheses that lead to a weak 'separation principle': that is, a theoretical justification for splitting the stochastic control problem in separate implementations of planning, localization, and control. In particular, the use of the information matrix decouples planning and localization: the planned paths will be safe whatever localization algorithm is used (Kalman filters, particle filters, etc.), as long as it is statistically efficient.
- We define our two algorithms as specializations of the same template, which is defined by a generic dominance relation '$\trianglerighteq$' and a generic precedence relation '$\blacktriangleleft$'. We believe this leads to clearer analysis and implementation.
- We extend the forward search algorithm first described in [1]. The robot is given the option to stay still for better localizing itself, and we use a more powerful dominance heuristics. Moreover, in addition to the minimum-time problem, we solve also the minimum-final-covariance problem.
- We describe another optimal algorithm for the minimum-time problem. It can be considered a back-projection algorithm: uncertainty constraints are propagated backward from the goal towards the start state. Its search tree can be re-utilized as long as the goal state remains the same. Interestingly, these two correct and complete algorithms often find distinct solutions.

Due to space constraints, proofs of the propositions are omitted; they can be found in [3]. The full C++ source code and datasets are available for download at the website `http://purl.org/censi/2007/ppu`. At the same site, there are animations of the search process in the *Flash* format.

II. Related work

There have been many approaches to robot motion planning with uncertainty. In this section, we first discuss the different formalizations used, and then we provide a cursory sample of the literature on the subject, in a more-or-less chronological order.

Which problem to solve. Different works attempted to solve different problems. This paper is concerned with two problems: minimizing the execution time, or minimizing the final covariance (with a bound on the execution time). Some approaches (e.g., [4], [5]) tried to solve a simpler problem: finding an admissible plan, without optimality properties. Other works ([6]) try to maximize the collected information, with free final pose. The approach in [7] maximizes the probability of success. Finally, many works do not explicitly model the uncertainty evolution in time, and only minimize a traversal cost, which is based on some "localizability" measure (e.g., [8]).

Representation of the uncertainty. There are two main options for representing uncertainty: a probabilistic representation, in terms of covariances, or a set-membership approach. In the set-membership approach ([5], [4]), the uncertainty is represented by a pose $\hat{\boldsymbol{q}}(t)$ and a set $\mathrm{SU}(t)$. The assumption is that the true pose $\boldsymbol{q}(t)$ belongs to the set obtained by enlarging $\hat{\boldsymbol{q}}(t)$ by the set $\mathrm{SU}(t)$: $\boldsymbol{q}(t) \in \hat{\boldsymbol{q}}(t) \oplus \mathrm{SU}(t)$ (here $\oplus$ denotes morphological dilatation).

A. Censi is with the Control & Dynamical Systems department, California Institute of Technology, 1200 E. California Blvd., 91125, Pasadena, CA. andrea@cds.caltech.edu

D. Calisi, A. De Luca, G. Oriolo are with the Dipartimento di Informatica e Sistemistica "A. Ruberti", Università di Roma "La Sapienza", via Ariosto 25, I-00185 Rome, Italy. {calisi,deluca,oriolo}@dis.uniroma1.it

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

	path planning	*stochastic control problem*	*path planning in extended space*
state space	poses	probability distribution on poses	poses×covariances
start	$\boldsymbol{q}^\star(0) = \boldsymbol{q}_{\text{start}}$ (1)	$\boldsymbol{q}(0) \sim$ assigned p.d. (2)	$\boldsymbol{q}_0^\star = \boldsymbol{q}_{\text{start}} \quad \boldsymbol{\Sigma}_0^\star = \boldsymbol{\Sigma}_{\text{start}}$ (3)
safety	$\boldsymbol{q}^\star(t) \in \mathcal{C}_{\text{free}}$ (4)	$\mathbb{P}(\boldsymbol{q}(t) \in \mathcal{C}_{\text{free}}) \geq 1 - \epsilon$ (5)	$\forall k : \forall \mathbf{M}_i \in \text{CONSTRAINTS}(\boldsymbol{q}_k^\star) : \ \boldsymbol{\Sigma}_k^\star \leq \mathbf{M}_i$ (6)
goal	$\boldsymbol{q}^\star(t_f) \in \mathcal{C}_{\text{target}}$ (7)	$\mathbb{P}(\boldsymbol{q}(t_f) \in \mathcal{C}_{\text{target}}) \geq 1 - \epsilon$ (8)	$\boldsymbol{q}_{t_f}^\star \in \mathcal{C}_{\text{target}} \quad \boldsymbol{\Sigma}_{t_f}^\star \leq \boldsymbol{\Sigma}_{\max}$ (9)
transition	Compatible with kynematic/dynamic constraints.	Given stochastic models of actions and sensing, the belief evolves according to Bayes' rule.	$\boldsymbol{q}_k \xrightarrow{\boldsymbol{\delta}} \boldsymbol{q}_k + \boldsymbol{\delta}$ (10) $\boldsymbol{\Sigma}_k \xrightarrow{\boldsymbol{\delta}} \left[\boldsymbol{\mathcal{I}}(\boldsymbol{q}_k, \boldsymbol{\delta}, \Delta t) + [\boldsymbol{\Sigma}_k + \boldsymbol{\Xi}(\boldsymbol{\delta})]^{-1}\right]^{-1}$ (11)
solution	A function $\boldsymbol{q}^\star(t)$.	A map from belief to actions.	A function $\boldsymbol{q}_k^\star$, which implies a certain $\boldsymbol{\Sigma}_k^\star$.
optimality	Minimizing a functional of $\boldsymbol{q}^\star(t)$.	Minimizing the *expectation* of a functional of $\boldsymbol{q}(t)$.	Two problems considered: 1) minimizing the final time 2) Minimizing the final covariance.

Fig. 1. Comparison of the three formalizations discussed in this paper: simple path planning (first column), the full stochastic control problem (second column), reduction of the stochastic control problem to path-planning in the extended space (third column).

The other option is to employ a probabilistic representation. The straightforward possibility is to propagate the mean/covariance of the distribution (e.g., [1], [9]). Some methods operate in a *reduced information space* ([10]), in which the belief is compressed by using a small set of parameters. For simplicity and computational advantages, some works use an isotropic representation of uncertainty ([4], [9]) — this might or might not be a limiting choice, depending on the other assumptions about the sensors and the environment.

Modeling of sensing actions. Modeling of the sensing actions can be done in a number of ways. One can have essentially blind robots, with only a limited form of odometry ([11]), or proximity sensors ([10]), or one may employ complex models of exteroceptive sensors ([12]). As for the exteroceptive sensors, one can choose to model the fact that a sensor is a source of continuous information, and the robot collects new information also by just standing still; this is the model used in this paper. Or, one could model an exteroceptive sensor as a device that allows a bounded uncertainty in its working zone, but this uncertainty does not decrease with time ([5]). Another sensing option is the use of "safe areas" (e.g., [13]). If the robot arrives in one of those areas, its uncertainty is reset to zero (or a small value), and it stays constant to that value while the robot remains in the area.

Plan representation. In general, the solution of a planning with uncertainty problem must be a "policy": a map from the history of observations to the action space. In some cases, the plan is a function from the *compressed* belief space to the actions [10]. In some other cases ([1]), the plan is a reference trajectory, along which the motion is guaranteed to be safe, in spite of uncertainty — this is also our case. Finally, the output of some methods ([13], [4]) is a 'sensor-based motion': a sequence of action/termination condition, where the condition depends on the sensory input ("walk until you see the wall").

Planning algorithm. The methods reviewed here have solved the problem in a variety of ways, covering most, if not all, known planning methods: back-projection [13], value/policy iteration [10], gradient descent [5], analytic closed-form solution [6], multi-sine optimization [14], classical graph search [1], dynamic programming [15], RRT [2], POMDPs [16].

In chronological order, the main family of approaches in the literature have been pre-image back-chaining [17], [13], [18]; sensory uncertainty fields [8], [19], [20], [21]; sensor-based planning [22], [4], [23]; the Information Space approach [10]; the set-membership approach [24], [5]; stochastic dynamic programming [15], [25]; and classical search strategies in an extended space [12], [1], [9], [26], [27]. The latter is the family to which this paper belongs. These methods try to solve the problem in the extended space of poses×covariances by applying classical search strategies, such as $A^\star$ and RRT.

III. Modeling motion planning with uncertainty

In this section, we show how taking into account uncertainty into a classic path-planning problem leads to a stochastic control problem; then we introduce the hypotheses and approximations that allow to reduce the problem to classic path-planning in the extended space of poses×covariances.

The table in Fig. 1 shows side-by-side the equations (1)–(11) used in the different formalizations. In the following, let $\boldsymbol{q}$ be the robot configuration. Bold uppercase letters refer to matrices: $\boldsymbol{\Sigma}$ is the covariance of the estimate of $\boldsymbol{q}$, $\boldsymbol{\mathcal{I}}$ is the Fisher information matrix.

A. From path-planning to the stochastic problem.

Path-planning can be formulated as minimizing a certain functional over a set of functions that respect constraints (1), (4), (7), plus any applicable kinematic and dynamic constraint. Note in particular that (1) and (7) impose that the initial and final states belong to prescribed sets, and that

(4) guarantees safety of the path by avoiding collisions with obstacles.

Uncertainty can be introduced by modeling the result of actions and sensing using stochastic equations. Instead of an initial state, one considers an initial probability distribution for the state. Now that the configuration is uncertain, one should replace the deterministic constraints (4)-(7) by the probabilistic constraints (5)-(8). Optimality criteria can be specified as the minimization of the *expectation* of a certain functional; for example, if in path-planning one minimizes the time to the goal, in the stochastic version one minimizes the *expected* time to the goal.

The solution of the stochastic control problem consists of two parts [28]: 1) an observer, which produces a probability distribution for the system state and 2) a control policy: a map from probability distributions to action.

The observer is the "easy" part, because, assuming the system is Markov, and given the state transition model $p(\boldsymbol{q}_k|\boldsymbol{q}_{k-1}, \boldsymbol{u}_k)$ and the sensor model $p(\boldsymbol{z}_k|\boldsymbol{q}_k)$, one can explicitly write the evolution of the belief $\mathrm{Bel}(\boldsymbol{q}_k)$:

$$\mathrm{Bel}(\boldsymbol{q}_k) \propto p(\boldsymbol{z}_k|\boldsymbol{q}_k)\int p(\boldsymbol{q}_k|\boldsymbol{q}_{k-1}, \boldsymbol{u}_k)\mathrm{Bel}(\boldsymbol{q}_{k-1})d\boldsymbol{q}_{k-1} \quad (12)$$

In this expression, $\boldsymbol{u}_k$ is the odometry data, and $\boldsymbol{z}_k$ is the sensory input. Of course, having an explicitly expression is often only of theoretical interest, because the equation is typically not solvable in closed form, except in the linear(ized) case, which corresponds to the (Extended) Kalman Filter. Particle filters solve the equation by approximating the integral by sampling [29].

The planning part is the "difficult" part: one must anticipate every possible belief, and even if the configuration space is finite-dimensional, the belief space is infinite-dimensional. There are algorithms that solve this problem in a generic way, typically discretizing the action and sensing space (policy iteration), but they suffer from the curse of dimensionality: the time required grows with the length of the plan and the number of possible actions and sensing events.

B. Simplifying the model

The goal of this section is to reduce the stochastic control problem to a deterministic problem in an extended space. The key to somehow decouple localization and planning is to be able to predict at planning time the uncertainty of the robot along a given path. This is only possible if the model is adequately simplified and additional hypotheses are needed on the environment and the localization algorithm.

In this paper, we are focusing on modeling the sensing aspect in a sound way, because it is the dominant part of the problem. As such, we ignore the dynamics and kinematics of the robot. In particular, we assume that an action $\boldsymbol{\delta}$ moves the robot configuration from $\boldsymbol{q}$ to $\boldsymbol{q} + \boldsymbol{\delta} + \boldsymbol{e}$, where $\boldsymbol{e}$ is a zero mean Gaussian variable with covariance $\boldsymbol{\Xi}(\boldsymbol{\delta})$. If also the distribution the sensor model $p(\boldsymbol{z}_k|\boldsymbol{q}_k)$ is Gaussian, then (12) transforms a Gaussian $\langle \hat{\boldsymbol{q}}_{k-1}, \boldsymbol{\Sigma}_{k-1}\rangle$ to another Gaussian $\langle \hat{\boldsymbol{q}}_k, \boldsymbol{\Sigma}_k\rangle$, with $\hat{\boldsymbol{q}}_k = \hat{\boldsymbol{q}}_{k-1} + \boldsymbol{\delta}$, and $\boldsymbol{\Sigma}_k$ given by:

$$\boldsymbol{\Sigma}_k = \left[\boldsymbol{\Sigma}^{-1}_{\boldsymbol{q}|\boldsymbol{z}} + \left[\boldsymbol{\Sigma}_{k-1} + \boldsymbol{\Xi}(\boldsymbol{\delta})\right]^{-1}\right]^{-1} \quad (13)$$

The term $\boldsymbol{\Sigma}^{-1}_{\boldsymbol{q}|\boldsymbol{z}}$ represents the information that the measurements $\boldsymbol{z}$ provide about $\boldsymbol{q}$. It is possible to estimate this term using the Cramér–Rao bound (CRB): $\boldsymbol{\Sigma}^{-1}_{\boldsymbol{q}|\boldsymbol{z}} \leq \boldsymbol{\mathcal{I}}(\boldsymbol{q}_k)$, where $\boldsymbol{\mathcal{I}}(\boldsymbol{q}_k)$ is the Fisher information matrix for the sensor — some regularity hypotheses are needed for the bound to be valid; we refer the reader to [30] for the technicalities. Therefore, the following bound (sometimes called the "Bayesian" Cramér–Rao bound) holds for the updated covariance:

$$\boldsymbol{\Sigma}_k \geq \left[\boldsymbol{\mathcal{I}}(\boldsymbol{q}_k) + \left[\boldsymbol{\Sigma}_{k-1} + \boldsymbol{\Xi}(\boldsymbol{\delta})\right]^{-1}\right]^{-1} \quad (14)$$

The inequality in (14) can be substituted by an equality if the CRB is strict and the localization algorithm is unbiased ($\mathrm{E}\{\hat{\boldsymbol{q}}\} = \boldsymbol{q}$) and statistically efficient.

Note that the information matrix depends on the sensor model, that is on the distribution $p(\boldsymbol{z}|\boldsymbol{q})$, and it depends on the actual pose $\boldsymbol{q}_k$, but it does *not* depend on the actual measurements. This means that it can be computed a priori, before taking the measurements, if one knows $\boldsymbol{q}_k$. But at planning time we just know our planned pose $\boldsymbol{q}^\star_k$: the best we can do is to assume $\boldsymbol{\mathcal{I}}(\boldsymbol{q}_k) \simeq \boldsymbol{\mathcal{I}}(\boldsymbol{q}^\star_k)$, that is the complexity of the environment (variability of $\boldsymbol{\mathcal{I}}$ in space) is low with respect to the localization and control errors that make up the difference $\boldsymbol{q} - \boldsymbol{q}^\star = (\boldsymbol{q} - \hat{\boldsymbol{q}}) + (\hat{\boldsymbol{q}} - \boldsymbol{q}^\star)$.

Let us recapitulate the assumptions we needed:

- The distribution of $\hat{\boldsymbol{q}}$ is well approximated by a Gaussian during the optimal motion.
- The localization algorithm is unbiased and statistically efficient. No other hypothesis is needed on the localization algorithm, that can be a (Extended) Kalman filter, a particle filter, etc.
- The uncertainty of the pose is low with respect to the environment complexity. We express this as a condition on the information matrix: $\boldsymbol{\mathcal{I}}(\boldsymbol{q}) \simeq \boldsymbol{\mathcal{I}}(\boldsymbol{q}^\star)$.

If these are satisfied, given a path $\{\boldsymbol{q}^\star_k\}$, we can recover the sequence of the covariance $\{\boldsymbol{\Sigma}^\star_k\}$ by using the transitions defined in (10), (11). Now, given a planned path $\{\langle \boldsymbol{q}^\star_k, \boldsymbol{\Sigma}^\star_k\rangle\}$, it is possible to evaluate whether it respects the safety constraints. Hence the stochastic control problem is reduced to path-planning in the extended space of poses $\times$ covariances.

In the following, we will define some other mundane details about the transitions, and how the safety constraints can be transformed into obstacles in the extended space.

C. Defining admissible states

From now on, we will indicate couples $\langle \boldsymbol{q}, \boldsymbol{\Sigma}\rangle$ as states. For characterizing admissible states, we note that (2), (5), (8), in the case of a polygonal world, are approximately equivalent to linear constraints on the covariance (Fig. 2). This model is similar to the one employed in [15].

We discretize the configuration space in cells, and for each cell $\boldsymbol{q}$ we define a set of matrices $\textsc{constraints}(\boldsymbol{q}) = \{\mathbf{M}_i\}$. The state $\langle \boldsymbol{q}, \boldsymbol{\Sigma}\rangle$ is admissible if $\boldsymbol{\Sigma} \leq \mathbf{M}_i$ for all $\mathbf{M}_i$ in $\textsc{constraints}(\boldsymbol{q})$. We use some of the elements of $\textsc{constraints}(\boldsymbol{q})$ to ensure safety of the motion (Fig. 2); but also arbitrary limits can be put in $\textsc{constraints}$ according to the application requirements. We set $\textsc{constraints}(\boldsymbol{q})$ to $\{\mathbf{0}\}$ (an impossible constraint to satisfy) if $\boldsymbol{q} \notin \mathcal{C}_{\text{free}}$.

D. Modeling sensor information and odometry uncertainty.

In our implementation, we assume that the odometry error due to a step $\boldsymbol{\delta}$ is proportional to the step size $|\boldsymbol{\delta}|$: $\boldsymbol{\Xi}(\boldsymbol{\delta}) = k|\boldsymbol{\delta}|\mathbf{I}$. The quantity of information acquired by the robot, during the motion from $\boldsymbol{q}$ to $\boldsymbol{q}+\boldsymbol{\delta}$ is assumed to be

$$\boldsymbol{\mathcal{I}}(\boldsymbol{q}, \boldsymbol{\delta}, \Delta t) = \Delta t \cdot \mathsf{sensor\_freq} \cdot (\boldsymbol{\mathcal{I}}(\boldsymbol{q}) + \boldsymbol{\mathcal{I}}(\boldsymbol{q}+\boldsymbol{\delta}))/2$$

where $\boldsymbol{\mathcal{I}}(\boldsymbol{q})$ is the information matrix for *one* sensor reading. Note that these expressions are valid also when the robot does not move ($\boldsymbol{\delta} = \mathbf{0}$): if the robot is in a zone where it senses something ($\boldsymbol{\mathcal{I}}(\boldsymbol{q}) \neq \mathbf{0}$), its uncertainty will decrease.

Ultimately, the matrix $\boldsymbol{\mathcal{I}}(\boldsymbol{q})$ depends on the particular sensor used. In the Section VII, we use range finders. For such sensors, the information matrix was studied in [31], along with matters of efficiency: it turns out that, in this case, the Cramér-Rao bound is a good approximation to the feasible localization accuracy.

E. Which problem to solve?

In this paper, we focus on two problems, of the many that could be defined in this framework:

Problem 1: PPU-T: Minimize the execution time, while remaining localized:

$$\min t_f, \qquad \text{subject to } \boldsymbol{\Sigma}(t) \leq \boldsymbol{\Sigma}_{\max}$$

Problem 2: PPU-COV: Minimize the final uncertainty, with a limit on the total time allowed:

$$\min_{\leq} \boldsymbol{\Sigma}(t_f) \qquad \text{subject to } t_f \leq t_{\max}$$

We refer to these two problems as 'PPU' (path planning with uncertainty), as opposed to 'PP' for simple path-planning. PPU has a number of interesting features.

Number of solutions: A solution for PPU might not exist even if it does for PP— for example, any case in which there is not enough information to remain localized. Also, it is common for PPU to have a continuum of solutions. For example, for PPU-COV, the final covariance depends on all the information acquired along the path, as a rather complicated non-linear functional of $\boldsymbol{\mathcal{I}}(\boldsymbol{q})$. Fix a path $\overline{\boldsymbol{q}}(t)$ with final covariance $\overline{\boldsymbol{\Sigma}}(t_f)$. In typical cases, small perturbations of $\overline{\boldsymbol{q}}(t)$ produce another final covariance $\tilde{\boldsymbol{\Sigma}}(t_f)$ such that neither $\tilde{\boldsymbol{\Sigma}}(t_f) \leq \overline{\boldsymbol{\Sigma}}(t_f)$ nor $\overline{\boldsymbol{\Sigma}}(t_f) \leq \tilde{\boldsymbol{\Sigma}}(t_f)$, and therefore the perturbed path is another, different, solution for the problem.

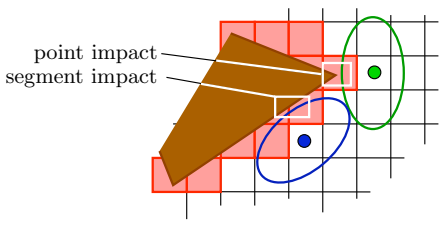


Fig. 2. A set of matrices CONSTRAINTS($\boldsymbol{q}$) is associated to each cell $\boldsymbol{q}$. The constraints are obtained by considering the possible collisions of the robot with the environment, due to its uncertainty.

Time is important: A property of PP is that, if one ignores dynamic constraints, time is merely one of the possible parameterizations for the path; instead, the right timing is essential for PPU, as sensors have a well-defined acquisition frequency: a given path could be feasible only if performed slowly enough, as a faster pace would result in less sensor readings than needed to remain localized.

Another feature of PPU is that the solution might include moments when the robot stays still to acquire more information at a particular point; in general, the projection to $\boldsymbol{q}$ of the $\boldsymbol{q} \times \boldsymbol{\Sigma}$ trajectory might contain loops.

IV. GENERIC SEARCH FRAMEWORK

All the algorithms described in the following sections are specializations of the same generic template (see, for example, [27]), enriched by a dominance relation $\trianglerighteq$, which is used to discard useless nodes. To instantiate the template, one needs to define the state space, the function SUCC($\boldsymbol{n}$) which generates the successors of $\boldsymbol{n}$, the predicate IS_GOAL, and the two dominance and precedence relations $\trianglerighteq$ and $\blacktriangleleft$.

Algorithm 1 Generic search algorithm

```
1:  VISITED: a ▷=-poset of length 1.
2:  OPEN: a ▷=-poset of length 1, ordered by ◀.
3:  Put n_0 in OPEN.
4:  while OPEN is not empty do
5:    Pop first (according to ◀) node n from OPEN.
6:    for all s in SUCC(n) do
7:      Report success if IS_GOAL(s).
8:      Ignore s if it is ▷=-dominated in VISITED.
9:      Discard nodes in VISITED ▷=-dominated by s.
10:     Put s in VISITED.
11:     Discard nodes in OPEN ▷=-dominated by s.
12:     Put s in OPEN.
13:   end for
14: end while
15: Report failure.
```

The partial order[1] $\trianglerighteq$ is used to define node "dominance" and therefore discard nodes. If nodes $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are present, and $(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2)$ is true, then $\boldsymbol{n}_2$ is discarded. Algorithm 1 guarantees that, at any moment, no node dominates any other, in either OPEN or VISITED – both sets are posets[2] of length 1.

The total order[3] $\blacktriangleleft$ is used to order the list of open nodes. In a totally ordered set, there is always a "minimum"; this property is needed in Algorithm 1 at line 5, in which the "first" element is extracted from OPEN.

V. FORWARD SEARCH

This section describes a forward search algorithm in the extended space of poses×covariances. We describe it by defining the variable parts of the generic template.

[1] A partial order is a reflexive, antisymmetric, and transitive relation.

[2] A poset is a set with a partial order defined on it. The length of a poset is the size of the longest chain $\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2 \trianglerighteq \ldots \trianglerighteq \boldsymbol{n}_n$.

[3] A total order is a partial order where any two elements are comparable.

Nodes. A node $\boldsymbol{n}$ is a tuple $\langle \boldsymbol{q}, \boldsymbol{\Sigma}, t \rangle$, whose intuitive meaning is: "It is possible to go from start to $\boldsymbol{q}$ in time t with final covariance $\boldsymbol{\Sigma}$.". Successors to a node are created using (10), (11).

Definition of the '$\trianglerighteq$' relation. When a search algorithm like $A^\star$ is used to solve PP, a dominance relation is implicitly defined. In PP, nodes are couples containing the pose and the time $\langle \boldsymbol{q}, t \rangle$; a node is discarded if it reaches the same pose of another one, but in more time (cost). This is the resulting dominance relation for PP: $(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \Leftrightarrow (\boldsymbol{q}_1 = \boldsymbol{q}_2) \wedge (t_1 \leq t_2)$. The direct extension of this to PPU is $(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \Leftrightarrow (\boldsymbol{q}_1 = \boldsymbol{q}_2) \wedge (t_1 \leq t_2) \wedge (\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2)$; better yet is to exploit the $\leq$ relation for matrices[4], by defining

$$(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \Leftrightarrow (\boldsymbol{q}_1 = \boldsymbol{q}_2) \wedge (t_1 \leq t_2) \wedge (\boldsymbol{\Sigma}_1 \leq \boldsymbol{\Sigma}_2) \quad (15)$$

In simple PP, '$\trianglerighteq$' is a *total* order: all the nodes with the same $\boldsymbol{q}$ are comparable according to their cost (time), therefore there is, at most, one node per cell. This is not true in PPU, as $\trianglerighteq$ is only a partial order on nodes: it is common that during the search there will be many nodes for the same cell because their covariances are not comparable (neither $\boldsymbol{\Sigma}_1 \leq \boldsymbol{\Sigma}_2$ nor $\boldsymbol{\Sigma}_2 \leq \boldsymbol{\Sigma}_1$, as in Fig. 3, left), or one has better time, but worse covariance (Fig. 3, right).

Trading time for information. The '$\trianglerighteq$' relation can be further extended to possibly discard a node in the latter case. Let the two nodes $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ have the same pose ($\boldsymbol{q}_1 = \boldsymbol{q}_2$), with $\boldsymbol{n}_1$ having better time ($t_1 < t_2$) and $\boldsymbol{n}_2$ having better covariance ($\boldsymbol{\Sigma}_2 \leq \boldsymbol{\Sigma}_1$). Consider node $\boldsymbol{n}_1$: to dominate $\boldsymbol{n}_2$, it needs to compensate the difference in the covariance in a time $\Delta t = (t_2 - t_1)$. If the environment is such that $\boldsymbol{\mathcal{I}}(\boldsymbol{q}_1) \neq \mathbf{0}$, the robot might remain still and acquire measurements to decrease its covariance. After waiting for Δt, its covariance will be

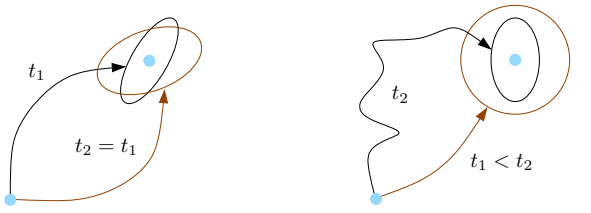
$$\text{WAIT}(\boldsymbol{q}_1, \boldsymbol{\Sigma}_1, \Delta t) \triangleq \left[\boldsymbol{\Sigma}_1^{-1} + \Delta t \cdot \mathsf{sensor\_freq} \cdot \boldsymbol{\mathcal{I}}(\boldsymbol{q}_1)\right]^{-1}$$

Therefore, there is a strategy (waiting) that generates the node $\boldsymbol{n}_1' = \langle \boldsymbol{q}_1, \text{WAIT}\left(\boldsymbol{q}_1, \boldsymbol{\Sigma}_1, (t_2 - t_1)\right), t_2 \rangle$ as a successor to $\boldsymbol{n}_1$. The node $\boldsymbol{n}_1'$ might, in turn, be comparable to $\boldsymbol{n}_2$, because it has the same time t_2. Hence, the node $\boldsymbol{n}_2$ can be discarded if $\text{WAIT}(\boldsymbol{q}_1, \boldsymbol{\Sigma}_1, t_2 - t_1) \leq \boldsymbol{\Sigma}_2$. Finally, we can update the definition of $\trianglerighteq$ as follows:

$$(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \quad \Leftrightarrow \quad (\boldsymbol{q}_1 = \boldsymbol{q}_2) \wedge (t_1 \leq t_2) \wedge \quad (16)$$

$$(\boldsymbol{\Sigma}_1 \leq \boldsymbol{\Sigma}_2 \vee \text{WAIT}(\boldsymbol{q}_1, \boldsymbol{\Sigma}_1, t_2 - t_1) \leq \boldsymbol{\Sigma}_2)$$

[4] $\mathbf{A} \leq \mathbf{B} \Leftrightarrow \mathbf{A} - \mathbf{B}$ is negative semidefinite.



(a) Same time, non comparable $\boldsymbol{\Sigma}$. (b) Better time but worse $\boldsymbol{\Sigma}$.

Fig. 3. Examples of nodes that are not comparable according to the relation defined by (15). Nodes in (b) *could* be comparable with the relation defined in (16).

Solving PPU-T. We solve the PPU-T problem using $A^\star$. In this case, '$\blacktriangleleft$' is defined as:

$$(\boldsymbol{n}_1 \blacktriangleleft \boldsymbol{n}_2) \quad \Leftrightarrow \quad t_1 + h(\boldsymbol{q}_1, \boldsymbol{q}_{\text{goal}}) \leq t_2 + h(\boldsymbol{q}_2, \boldsymbol{q}_{\text{goal}}) \quad (17)$$

In this expression $h(\boldsymbol{q}, \boldsymbol{q}_{\text{goal}})$ must be an 'admissible' heuristic. The choice of the heuristic is tied only to the cost function being minimized. Because the cost function (time spent) is the same in PP and PPU-T, the heuristic does not change, and the covariance plays no role in it.

Solving PPU-COV . We solve PPU-COV using wavefront expansion. With respect to the use of $A^\star$ in the last section, the relation $\trianglerighteq$ remains the same, while the relation $\blacktriangleleft$ and the termination condition will change. The relation $\blacktriangleleft$ is modified to expand nodes generation by generation in successive waves. The search does not terminate immediately if a state is found in $\boldsymbol{q}_{\text{goal}}$, because there could be a path with longer time but better covariance. At the end of the execution there will be, in general, more than one node at $\boldsymbol{q}_{\text{goal}}$, and all will have different times and covariances. At this point, one should take the minimal subset according to this relation: $(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \Leftrightarrow (\boldsymbol{\Sigma}_1 \leq \boldsymbol{\Sigma}_2)$: after the exploration, nodes with low times and big covariances (that *had* to be preserved), are not interesting anymore and can be safely discarded.

VI. Backward search

This section describes a search algorithm to solve the PPU-T problem, backward. In the previous section, *states* were propagated *forward* through *actions*, here *constraints* are propagated *backward* through the *inverse* of the actions. Unfortunately, because we represent uncertainty by a full matrix, instead of a single number, the computations get a bit tricky: in this paper, proofs and some details have been omitted and can be found in [3].

Nodes. A node is a tuple $\langle \boldsymbol{q}_k, \{\mathbf{M}_i\}, tg \rangle$, whose intuitive meaning is "There is a path from pose $\boldsymbol{q}_k$ to the goal, in time tg (time-to-goal), if the covariance is such that $\forall i : \ \boldsymbol{\Sigma} \leq \mathbf{M}_i$". The first node to be put into the OPEN list is the node relative to $\boldsymbol{q}_{\text{goal}}$ and the constraints to be respected there: $\boldsymbol{n}_0 = \langle \boldsymbol{q}_{\text{goal}}, \text{CONSTRAINTS}(\boldsymbol{q}_{\text{goal}}), 0 \rangle$: this node represents the goal states.

Successors. Consider a node $\langle \boldsymbol{q}_k, \{\mathbf{M}_i\}, tg \rangle$. For any possible forward action $\boldsymbol{\delta}$, the successor to $\boldsymbol{n}$ are generated by applying the inverse action $-\boldsymbol{\delta}$. The successor to the node will be $\boldsymbol{n}_{-\boldsymbol{\delta}} = \langle \boldsymbol{q}_k - \boldsymbol{\delta}, \ \cdot \ , tg + \Delta t \rangle$. The set of constraints will be transformed from $\{\mathbf{M}_i\}$ to

$$\{\mathbf{M}_i\} \overset{-\boldsymbol{\delta}}{\rightarrow} \{\text{back-proj}(\mathbf{M}_i)\} \ \cup \ \text{CONSTRAINTS}(\boldsymbol{q}_k - \boldsymbol{\delta}) \quad (18)$$

The back-projection of a single constraint $\mathbf{M}_i$ is that matrix back-proj($\mathbf{M}_i$) such that $\boldsymbol{\Sigma}_{k-1} \leq \text{back-proj}(\mathbf{M}_i)$ is equivalent to $\boldsymbol{\Sigma}_k \leq \mathbf{M}_i$. By algebraically manipulating the covariance update equation (11), one obtains that $\boldsymbol{\Sigma}_k \leq \mathbf{M}_i$ is equivalent to the following:

$$\boldsymbol{\Sigma}_{k-1} \leq \left[\mathbf{M}_i{}^{-1} - \boldsymbol{\mathcal{I}}(\boldsymbol{q}, \boldsymbol{\delta}, \Delta t)\right]^{-1} - \boldsymbol{\Xi}(\boldsymbol{\delta})$$

and therefore one can define back-proj$(\mathbf{M}_i)$ as $\left[\mathbf{M}_i{}^{-1} - \boldsymbol{\mathcal{I}}(\boldsymbol{q}, \boldsymbol{\delta}, \Delta t)\right]^{-1} - \boldsymbol{\Xi}(\boldsymbol{\delta})$.

However, there are many special cases that must be taken care of. It could happen that the constraint is trivial to satisfy: this is the case when $\left[\mathbf{M}_i{}^{-1} - \boldsymbol{\mathcal{I}}(\boldsymbol{q}, \boldsymbol{\delta}, \Delta t)\right] \leq 0$; if the constraint is trivial, it can be removed from the constraint set. It could happen that the constraint is impossible to satisfy: this is the case when at least one eigenvalue of back-proj$(\mathbf{M}_i)$ is negative; if the constraint cannot be satisfied, the successor is discarded.

Simplification of the constraints set. A recursive application of (18) make the number of constraints grow indefinitely, as it "collects" geometry constraints along the reverse path. Therefore, it is important to find criteria to simplify this set of constraints, when possible. In practice, the following criteria allow to keep the number of constraints as low as 3-4 per node in environments similar to that used in Section VII.

In the following, let $\textsc{resp}(\{\mathbf{M}_i\})$ be the set of covariance matrices that respects all the constraints $\{\mathbf{M}_i\}$:

Definition 1: $\textsc{resp}(\{\mathbf{M}_i\}) \triangleq \{\boldsymbol{\Sigma} \,|\, \forall i : \boldsymbol{\Sigma} \leq \mathbf{M}_i\}$

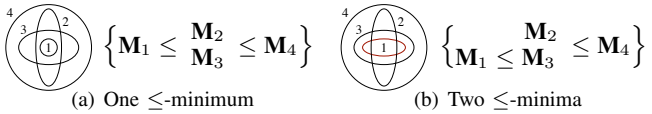
Formally, we define the simplification problem as

Problem 3: (Constraints simplification) Given a set $\{\mathbf{M}_i\}$, find a smaller set $\{\mathbf{N}_j\}$ such that $\textsc{resp}(\{\mathbf{M}_i\}) = \textsc{resp}(\{\mathbf{N}_j\})$.

From the following proposition, we conclude that, in order to simplify the constraints, we only need to discard some matrices from $\{\mathbf{M}_i\}$.

Proposition 1: If two constraints sets $\{\mathbf{M}_i\}$ and $\{\mathbf{N}_j\}$ define the same admissible set of covariances ($\textsc{resp}(\{\mathbf{M}_i\}) = \textsc{resp}(\{\mathbf{N}_j\})$) , then one is a subset of the other.

A first idea is for the simplification is to exploit the partial order induced by $\leq$ on the matrices. In the following example (a), the set $\{\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4\}$ can be reduced to $\{\mathbf{M}_1\}$. This case is lucky because there exists only one $\leq$-minimum. In case (b), there are two minima, $\{\mathbf{M}_1, \mathbf{M}_2\}$, which must be kept.



(a) One $\leq$-minimum (b) Two $\leq$-minima

Using covariance intersection. There is, however, a more powerful heuristics using covariance intersection. *Covariance Intersection* (CI) is a statistical tool that allows to perform data-fusion of measurements having unknown correlation [32].

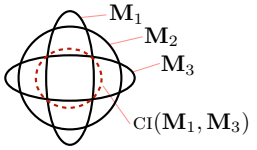
Definition 2: Given a set of positive definite matrices $\{\mathbf{M}_i\}$, and a vector $\boldsymbol{\omega}$ such that $\sum_i \omega_i = 1$, define the covariance intersection as $\textsc{ci}(\{\mathbf{M}_i\}, \boldsymbol{\omega}) \triangleq \left[\sum \omega_i \mathbf{M}_i^{-1}\right]^{-1}$.

Here, we use only a *geometric* property of CI:

Proposition 2: A matrix $\boldsymbol{\Sigma}$ belongs to $\textsc{resp}\left(\{\mathbf{M}_i\}\right)$ iff it is smaller than the covariance intersection of $\{\mathbf{M}_i\}$ for all values of $\boldsymbol{\omega}$:

$$\boldsymbol{\Sigma} \in \textsc{resp}\left(\{\mathbf{M}_i\}\right) \;\Leftrightarrow\; \forall \boldsymbol{\omega} : \; \boldsymbol{\Sigma} \leq \textsc{ci}(\{\mathbf{M}_i\}, \boldsymbol{\omega})$$

The following is an example with three matrices. The three matrices do not dominate each other according to $\leq$, therefore they cannot be simplified using the previous criterion. But it is the case that $\textsc{ci}(\{\mathbf{M}_1, \mathbf{M}_3\}, 0.5) \leq \mathbf{M}_2$, and therefore $\mathbf{M}_2$ can be discarded.



With the CI trick, the simplification problem is reduced to the following.

Problem 4: Given a set $\{\mathbf{M}_i\}$ and a matrix $\mathbf{N}$, is there a vector $\boldsymbol{\omega}$ such that $\textsc{ci}(\{\mathbf{M}_i\}, \boldsymbol{\omega}) \leq \mathbf{N}$?

This is an open problem for us. In the implementation, when it is needed to check for this condition, the covariance intersection is done for only a finite number of values for $\boldsymbol{\omega}$: with only one element set to 1: $\boldsymbol{\omega} = \langle \ldots, 0, 1, 0, \ldots \rangle$; with two elements set to 0.5: $\boldsymbol{\omega} = \langle 0.5, 0.5, 0, \ldots \rangle$; with all elements set to $(1/n)$: $\boldsymbol{\omega} = \langle 1/n, \ldots, 1/n \rangle$.

Design of the '$\blacktriangleleft$' relation. To use an $A^\star$-like algorithm, define $\blacktriangleleft$ as

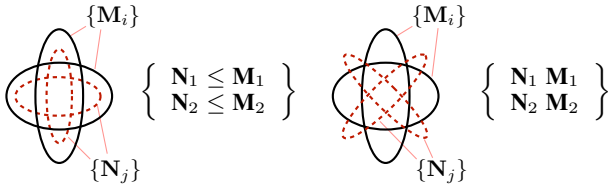
$$(\boldsymbol{n}_1 \blacktriangleleft \boldsymbol{n}_2) \quad \Leftrightarrow \quad tg_1 + h(\boldsymbol{q}_{\text{start}}, \boldsymbol{q}_1) \leq tg_2 + h(\boldsymbol{q}_{\text{start}}, \boldsymbol{q}_2)$$

Note that, this time, the heuristics must estimate the distance to the start, instead of to the goal, as in (17).

Design of the '$\trianglerighteq$' relation. A node $\boldsymbol{n}_1$ $\trianglerighteq$-dominates $\boldsymbol{n}_2$ if its time is better or equal, and all covariances admissible for $\boldsymbol{n}_2$ are also admissible for $\boldsymbol{n}_1$, that is, the set $\textsc{resp}(\{\mathbf{N}_j\})$ is a subset of $\textsc{resp}(\{\mathbf{M}_i\})$. Formally, the relation $\trianglerighteq$ is defined as

$$(\boldsymbol{n}_1 \trianglerighteq \boldsymbol{n}_2) \;\Leftrightarrow\; (t_1 \leq t_2) \;\wedge\; (\textsc{resp}(\{\mathbf{M}_i\}) \supseteq \textsc{resp}(\{\mathbf{N}_j\}))$$

Now we need a way to check whether $\textsc{resp}(\{\mathbf{M}_i\}) \supseteq \textsc{resp}(\{\mathbf{N}_j\})$. To start things off, if for all constraints in $\{\mathbf{M}_i\}$ there is a stricter constraint in $\{\mathbf{N}_j\}$, then the condition is met: this holds in the following example (left). However, the inverse implication does not hold: a counter-example is shown (right); in this case, $\textsc{resp}(\{\mathbf{N}_j\})$ is contained in $\textsc{resp}(\{\mathbf{M}_i\})$ but the matrices in $\{\mathbf{M}_i\}$ are not bigger than those in $\{\mathbf{N}_j\}$.

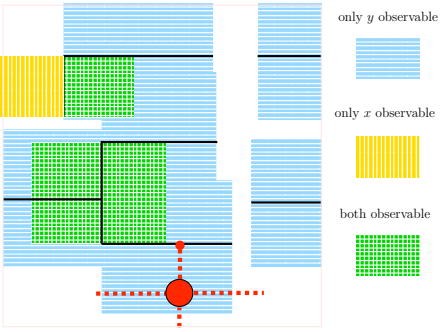


Covariance intersection gives, once again, a more powerful criterion:

Proposition 3: If for every matrix $\mathbf{M}_i$ there is a covariance intersection of the matrices $\{\mathbf{N}_j\}$ which is smaller than $\mathbf{M}_i$, then $\textsc{resp}(\{\mathbf{N}_j\})$ is contained in $\textsc{resp}(\{\mathbf{M}_i\})$.

$$\forall i \, \exists \boldsymbol{\omega} : \textsc{ci}(\{\mathbf{N}_j\}, \boldsymbol{\omega}) \leq \mathbf{M}_i \Rightarrow (\textsc{resp}(\{\mathbf{M}_i\}) \supseteq \textsc{resp}(\{\mathbf{N}_j\}))$$

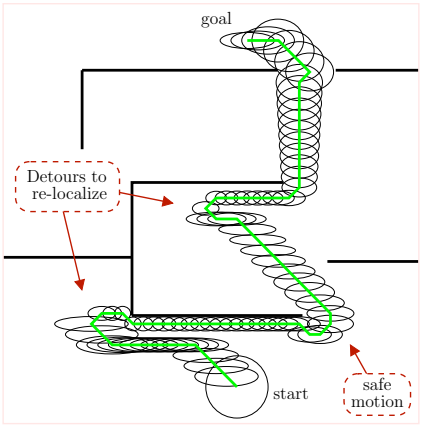
Therefore, also the design of the $\trianglerighteq$ relation is reduced to solving Problem 4.



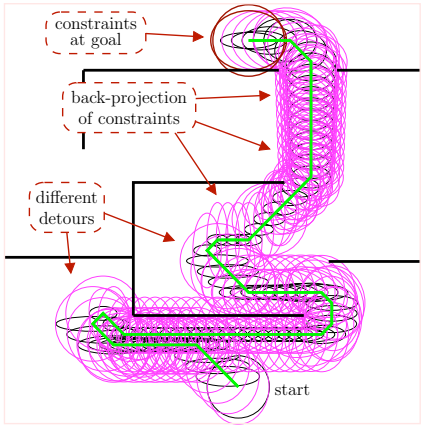
(c) Information in the environment

	PPU-T forward $A^\star$	PPU-T backward $A^\star$	PP forward $A^\star$	PPU-COV forward wave	PP forward wave
created nodes	5 474	10 110	369	11 956	1 943
popped nodes	4 211	7 321	229	9 830	1 536
active nodes	4 477	7 337	328	9 767	1 500
epochs	-	-	-	109	58
matrix comparisons	106 252	1 021 156	-	233 874	-
time - G4 (seconds)	0.51	2.13	0.04	5.01	0.10
time - P4 (seconds)	0.23	1.06	0.02	2.50	0.05

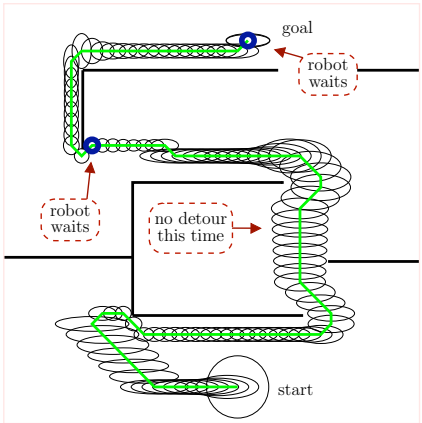
(d) Performance comparison



(e) Minimum-time solution (forward algorithm)



(f) Minimum-time solution (backward algorithm)



(g) Minimum-final-covariance solution

Fig. 4. Legend for the table: *created nodes:* Number of nodes that entered the OPEN or VISITED set. This number does not include nodes that got created by SUCC but were immediately discarded because they were dominated; *popped nodes:* number of nodes that were popped out of OPEN and then expanded; *active nodes:* Number of nodes contained in VISITED when the simulation stopped; *epoch:* Number of generations expanded; *matrix comparisons* number of times that it was needed to check for a matrix inequality; *time - G4 1.5GHz*: CPU time needed on a PowerPC G4 1.5GHz; *time - P4 2.8GHz*: CPU time needed with a Pentium 4 Xeon 2.8GHz. In both cases, the program was compiled with GCC 3.3 with options `-O2`.

VII. Examples

For illustrative purposes, we show here the results for a simple environment/sensor for which the three algorithms give very different results. We assume the robot to have 4 range sensors mounted in direction N,S,W,E, and the walls are only in the N-S or W-E directions: in this way, the environment is partitioned in four kinds of zones (Fig. 4(c)): where the robots receives no sensing, x observable, y observable, both x and y observable. For this configuration, the resulting paths are easily interpreted.

Fig. 4(e) and 4(f) show the PPU-T solutions found by the forward and backward algorithm. They have the same time (up to the cell discretization), but the paths are different. In fact, PPU-T has, in general, a continuum of solutions. The typical case is that the robot must make a detour to relocalize: in Fig. 4(e)-4(f) one can see the detour about half-way along the path; choosing when to make this detour produces different solutions. It is common that the two algorithms produce different (optimal) solutions, as their search is biased in different ways.

Fig. 4(g) shows the solution for PPU-COV found by the forward wavefront-expansion algorithm; this solution is topologically different from the PPU-T solution. Here, there are two points where the robot stays still to acquire more information: in the middle of the path, to relocalize before a difficult passage, and at the end: when the robot arrives at the goal, the best option is to stay still and acquire information until the available time expires.

From the table in Fig. 4(d), we see that the backward search expands more nodes than the forward search (about twice as much) and takes more time (it is about four times slower). However, because they start their search at different places, it is easy to construct examples where any of the two perform very badly with respect to the other.

Also note that the backward search provide more information: its search tree can be reused for the next query if the goal state remains the same. Moreover, the output of the backward algorithm contains the bounds $\{\mathbf{M}_i\}$ at each step (the purple ellipses in Fig. 4(f)): if, for any reason, during the motion, the covariance of the estimate is bigger than the planned covariance — e.g., because of unexpected occlusions in the environment — one can compare the new covariance to the bounds in the plan: if the bounds are still respected, than the plan is still valid and still optimal.

On the website `http://purl.org/censi/2007/ppu`, there are Flash animations of the search process for all methods, for this and other environments.

VIII. Conclusions and future work

In this paper, we modeled motion planning with uncertainty using a Bayesian framework. We sought hypotheses that justify a separate implementation of localization, control, and planning. Under those hypotheses, the intractable stochastic control problem can be reduced to path-planning in the extended space of poses$\times$covariances.

Unfortunately, this fine formalization leads to complicated computations. The problem is that the set of covariances is only partially ordered by the relation $\leq$. If one used a one-parameter representation, as in [26], the relation '$\leq$' would be a total ordering, and one could throw away all the exotic propositions in Section VI. Nevertheless, we believe it is worth keeping this formalization, as it is the same currently used in localization/SLAM research.

We considered two problems: minimizing the final time, and minimizing the final covariance. For the first problem we proposed two algorithms, both optimal, which are defined as two different specializations of the same generic template.

The first algorithm is a classic forward search. With respect to previous work, we allow the robot to stay still, and we formulate a more powerful dominance relation.

The second algorithm propagates constraint sets from the goal towards the start. For a single query, it is slower in typical cases, but its search tree can be re-utilized as long as the goal remains the same. The fact that these two optimal algorithms often give different solutions is an experimental validation that the problem admits a continuum of solutions.

The second problem, minimizing the final covariance, is the one that we think will provide a bridge to the SLAM problem. For an effective exploration strategy, the robot must be well localized in the known map, before sensing the unknown territory; so PPU-COV could be seen as a sub-problem of optimal planning for SLAM. In this paper, we solved the PPU-COV in a very naive way, by using wavefront expansion. At the moment, it is not clear to us how to design an effective $\blacktriangleleft$ relation which will lead to a more efficient search: that is part of future work.

References

[1] A. Lambert and D. Gruyer, "Safe path planning in an uncertain-configuration space," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, vol. 3, pp. 4185–4190, Sept. 2003.

[2] R. Pepy and A. Lambert, "Safe path planning in an uncertain-configuration space using RRT," in *Proc. of the IEEE/RSJ Intl. conf. on Intelligent Robots and Systems*, pp. 5376–5381, Oct. 2006.

[3] A. Censi, "Robot motion planning with uncertainty," Master's thesis, Università di Roma 'La Sapienza', May 2007.

[4] B. Bouilly, T. Siméon, and R. Alami, "A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, 1995.

[5] L. A. Page and A. C. Sanderson, "A path-space search algorithm for motion planning with uncertainties," in *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, (Pittsburgh, PA, USA), pp. 334–340, Aug. 1995.

[6] F. Lorussi, A. Marigo, and A. Bicchi, "Optimal exploratory paths for a mobile rover," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, pp. 2078–2083, 2001.

[7] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty," in *Proceedings of Robotics: Science and Systems*, (Atlanta, GA, USA), June 2007.

[8] H. Takeda and J.-C. Latombe, "Sensory Uncertainty Field for mobile robot navigation," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, (Nice, France), pp. 2465–2472, May 1992.

[9] J. P. Gonzalez and A. T. Stentz, "Planning with uncertainty in position: An optimal and efficient planner," in *Proc. of the IEEE/RSJ Intl. conf. on Intelligent Robots and Systems*, pp. 2435 – 2442, August 2005.

[10] J. Barraquand and P. Ferbach, "Motion planning with uncertainty: The Information Space approach," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, 1995.

[11] J. M. O'Kane, "Global localization using odometry," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, 2006.

[12] A. Lambert and N. L. Fort-Piat, "Safe task planning integrating uncertainties and local maps federations," *International Journal of Robotics Research*, vol. 19, pp. 597–611, Jun 2000.

[13] A. Lazanas and J.-C. Latombe, "Landmark-based robot navigation," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, (San Jose, California), pp. 816–822, AAAI Press, 1992.

[14] L. Mihaylova, J. D. Schutter, and H. Bruyninckx, "A multisine approach for trajectory optimization based on information gain," in *Proc. of the IEEE/RSJ Intl. conf. on Intelligent Robots and Systems*, (Lausanne, Switzerland), 2002.

[15] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[16] K. Hsiao, L. Kaelbling, and T. Lozano-Pérez, "Grasping pomdps," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, pp. 4685–4692, 2007.

[17] T. Lozano-Pérez, M. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, 1984.

[18] T. Fraichard and R. Mermond, "Path planning with uncertainty for car-like robots," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, pp. 27–32, 1998.

[19] P. E. Trahanias and Y. Komninos, "Robot motion planning: Multi-Sensory Uncertainty Fields enhanced with obstacle avoidance," in *Proc. of the IEEE/RSJ Intl. conf. on Intelligent Robots and Systems*, 1996.

[20] N. A. Vlassis and P. Tsanakas, "A Sensory Uncertainty Field model for unknown and non-stationary mobile robot environments," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, 1998.

[21] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[22] R. Alami and T. Siméon, "Planning robust motion strategies for a mobile robot," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, 1994.

[23] M. Khatib, B. Bouilly, T. Siméon, and R. Chatila, "Indoor navigation with uncertainty using sensor-based motions," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, vol. 4, (Albuquerque, NM, USA), pp. 3379–3384, Apr. 1997.

[24] L. A. Page and A. C. Sanderson, "Robot motion planning for sensor-based control with uncertainties," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, vol. 2, (Nagoya, Japan), pp. 1333–1340, May 1995.

[25] L. Blackmore, "A probabilistic particle control approach to optimal, robust predictive control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[26] J. P. Gonzalez and A. Stentz, "Planning with uncertainty in position using high-resolution maps," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, (Rome, Italy), April 2007.

[27] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[28] J. M. O'Kane, B. Tovar, P. Cheng, and S. M. LaValle, "Algorithms for planning under uncertainty in prediction and sensing," in *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, Series in Control Engineering, ch. 13, Marcel Dekker, 2006.

[29] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[30] H. L. V. Trees and K. L. Bell, *Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking*. Wiley-IEEE Press, 2007.

[31] A. Censi, "On achievable accuracy for range-finder localization," in *Proc. of the IEEE Intl. conf. on Robotics & Automation*, (Rome, Italy), pp. 4170–4175, April 2007.

[32] J. Uhlmann, S. Julier, and M. Csorb, "Nondivergent simultaneous map building and localization using covariance intersection," in *Proceedings of the SPIE Aerosense Conference*, pp. 3087–, Apr 1997.