

# Global Estimation in Constrained Environments

Marin Kobilarov, Jerrold E. Marsden, Gaurav S. Sukhatme

marin@cds.caltech.edu, marsden@cds.caltech.edu, gaurav@usc.edu

Original submission: December 31, 2009

Last revision: August 16, 2011

## Abstract

This article considers the optimal estimation of the state of a dynamic observable using a mobile sensor. The main goal is to compute a sensor trajectory which minimizes the estimation error over a given time horizon taking into account the uncertainty in the observable dynamics and its sensing and respecting the constraints of the workspace. The main contribution is a methodology for handling arbitrary dynamics, noise models and environment constraints in a global optimization framework. It is based on sequential Monte Carlo methods and sampling-based motion planning. Three variance reduction techniques—utility sampling, shuffling, and pruning—based on importance sampling, are proposed to speed-up convergence. The developed framework is applied to two typical scenarios: a simple vehicle operating in a planar polygonal obstacle environment; a simulated helicopter searching for a moving target in a 3-D terrain.

## 1 Introduction

Consider a mobile sensor (a *vehicle*) estimating the state of an observable (a *target*) with uncertain dynamics through noisy measurements. The vehicle and target motions are constrained due to their natural kinematics and dynamics and due to obstacles in the environment. The task is to compute an open-loop vehicle trajectory over a given time horizon resulting in a target state estimate with lowest uncertainty. Such a capability arises, for instance, in the context for time-critical *surveillance* or *search-and-rescue* missions.

The problem is formally defined through a hidden Markov model (HMM) of a stochastic process  $\{(X_k, Y_k)\}_{0 \leq k \leq N}$  where  $X_k$  denotes the hidden target state and  $Y_k$  denotes the observation at the  $k$ -th time epoch. The process evolution is studied over a horizon of  $N$  epochs. The respective state and observation realizations are denoted by  $x_k \in \mathcal{X} \subset \mathbb{R}^{n_x}$  and  $y_k \in \mathcal{Y} \subset \mathbb{R}^{n_y}$ , where  $\mathcal{X}$  and  $\mathcal{Y}$  are vector spaces. The HMM is defined according to

$$X_{k+1} = f(X_k, \Omega_k), \quad (1a)$$

$$Y_k = g(X_k, V_k; \mu_k), \quad (1b)$$

where  $\Omega_k$  and  $V_k$  are i.i.d. noise terms and  $\mu_k \in \mathcal{M}$  denotes the vehicle state<sup>1</sup> at time  $k$ . The manifold  $\mathcal{M}$  need not be a vector space. A trajectory of states between two epochs  $i$  and  $j$ , where  $0 \leq i < j \leq N$ , is denoted by  $x_{i:j}$ , i.e.  $x_{i:j} := \{x_i, x_{i+1}, \dots, x_{j-1}, x_j\}$ .

---

<sup>1</sup>a *state* denotes both the configuration and velocity of the vehicle (i.e. “vehicle state”) or the target (i.e. “target state”); when used simply as “state” its meaning will be clear from the context.

The vehicle trajectory  $\mu_{0:N}$  is subject to dynamical constraints, e.g. arising from discretized Euler-Lagrange equations of motion, expressed through the equality  $h_d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}^{n_d}$

$$h_d(\mu_k, \mu_{k+1}) = 0, \text{ for all } 0 \leq k < N. \quad (2)$$

In addition the vehicle must avoid obstacles and is subject to velocity and actuator bounds, jointly encoded through the inequality constraints  $h_c : \mathcal{M} \rightarrow \mathbb{R}^{n_c}$

$$h_c(\mu_k) \geq 0, \text{ for all } 0 \leq k \leq N. \quad (3)$$

A trajectory that satisfies the constraints (2) and (3) is termed *feasible*.

The functions  $h_d$  and  $h_c$  are typically *non-convex* and in some cases *non-smooth*. On one hand, complicated nonlinear dynamics and obstacles induce multiple homotopy classes of vehicle trajectories that preclude convexity. On the other, the function (2) can have singularities (and hence be not everywhere smooth) when the dynamics is underactuated or nonholonomic (Choset et al., 2005; LaValle, 2006). In addition, the distance-to-obstacle function (encoded in (3)) might not be differentiable e.g. at sharp obstacle corners, requiring either ad-hoc smoothing or special non-smooth techniques (Clarke et al., 1998) such as *generalized gradients* (Choset et al., 2005).

The vehicle computes numerically the target state distribution, also referred to as the *filtering distribution*, denoted by  $\pi_k(dx|y_{1:k}; \mu_{0:k})dx := \mathbb{P}(X_k \in dx|y_{1:k}; \mu_{0:k})$  with respect to some standard measure  $dx$  assuming the vehicle has moved along trajectory  $\mu_{0:k}$  and obtained a sequence of measurements  $y_{1:k}$ .

**Objective.** The goal is to control the vehicle to obtain a high-quality estimate of the target state during the *future*  $N$  epochs. Typically, only a subset of the target coordinates are of interest. An appropriately chosen function  $\varphi : \mathcal{X} \rightarrow \mathbb{R}^{n'_x}$ , where  $n'_x \leq n_x$ , selects and weighs a combination of these coordinates. For instance,  $\varphi$  can pick out only the position of a moving target and ignore its velocity and heading. The optimization problem is to compute the optimal future vehicle trajectory  $\mu_{1:N}^*$  which minimizes the target estimate *uncertainty* defined through

$$\mu_{1:N}^* = \arg \min_{\mu_{1:N}} \mathbb{E} [\|\varphi(X_N) - \int \varphi(x)\pi_N(x|Y_{1:N}; \mu_{1:N})dx\|^2], \quad (4)$$

subject to the dynamics (2) and constraints (3). The expectation in (4) is taken over all future realization of the states  $X_{0:N}$  and the measurements  $Y_{1:N}$  while  $\pi_N$  is the posterior density after filtering these measurements given (1).

The cost function in (4) is equivalent to the *trace of the covariance* of  $\varphi(X_N)$ . While it is possible to use other measures such as entropy or covariance determinant, this metric is chosen since its value can be interpreted in meaningful units (e.g. see Mihaylova et al., 2003a). For instance, the special case  $\varphi(x) = M^{\frac{1}{2}}x$  for some weighing matrix  $M$  corresponds to a well-established tolerance-weighted error or L-optimal design (de Geeter et al., 1998).

**Simple Example.** These definitions can be illustrated with a simple example of a target modeled as a unit mass particle moving in the plain. A vehicle with fixed constant velocity  $v \in \mathbb{R}^2$  takes relative position measurements and must avoid a circular obstacle with center  $o \in \mathbb{R}^2$  and radius  $d$ . In this situation we have  $\mathcal{X} = \mathcal{Y} = \mathcal{M} = \mathbb{R}^2$  with motion function  $f(x, \omega) = x + \omega$ , observation function  $g(x, \nu; \mu) = x - \mu + \nu$ , and vehicle dynamics and constraints given by  $h_d(\mu_k, \mu_{k+1}) = \mu_{k+1} - \mu_k - v$  and  $h_c(\mu) = \|\mu - o\| - d$ , respectively. The noise terms  $\omega$  and  $\nu$  are realizations of e.g. white random processes. Setting  $\varphi(x) = x$  is then equivalent to minimizing the sum of the variances of the two planar coordinates. Therefore, if  $x$  were measured in  $m$  (meters) then the square root of the right hand side of (4) is also in  $m$  which is convenient for establishing meaningful tolerances.

**Related Work.** The optimization (4) corresponds to the *optimal sensor scheduling* problem (Tremois and Le Cadre, 1999; Singh et al., 2007) which is of central importance in the target tracking community. It is also highly relevant to the problem of *active sensing* studied in robotics (Thrun et al., 2005; Grocholsky et al., 2003; Mihaylova et al., 2003a) where the vehicle is estimating its own state (Paris and Le Cadre, 2002; He et al., 2008) and in some cases refining its knowledge about the environment (Stachniss et al., 2005; Sim and Roy, 2005).

One approach is to solve the problem approximately by discretizing the vehicle and target state spaces. Such techniques, e.g. based on regular grids (e.g. Chung and Burdick, 2007) or shaping functions (Lavis et al., 2008), are too restrictive when non-trivial dynamics and sensing are considered. They are more appropriate for higher-level decision making. For instance, policy search in information spaces (LaValle, 2006) or Markov decision process (MDP)-based search (e.g. (Bethke et al., 2008)) are typically based on such representations. We emphasize that our main interest is in high-dimensional problems dominated by fast nonlinear and underactuated dynamics and sensing. In this context techniques exploiting convexity (resp. sub-modularity (Krause and Guestrin, 2007; Hollinger et al., 2009)) are not suitable since such approximations are either too coarse or will violate the dynamics and result in solutions that the original system cannot realistically execute.

Most existing techniques, beyond discrete methods, have one or more of the following limitations: they are based on models with linear/Gaussian structure; are limited to myopic one-step optimal decision making; the state space is unconstrained (i.e. no constraints of the form (3) are considered). Several recent works have addressed some of these issues but not all. For instance, simulation-based stochastic gradient optimization is proposed by (Singh et al., 2007) in order to handle arbitrary motion and sensor models. The resulting method is provably convergent and exploits problem structure through control variates to reduce variance. A related approach (Martinez-Cantin et al., 2009) aimed at on-line active sensing employs Bayesian optimization, i.e. using Gaussian process cost function approximation to speedup the search. Several recent works with application to unmanned aerial vehicles (UAVs) also address some of the listed limitations but are still restricted to either stationary targets (Tisdale et al., 2009), one-step planning (Cole et al., 2008; Hoffmann and Tomlin, 2009; Bryson and Sukkarieh, 2009), or unconstrained scenarios (Ryan, 2008; Geyer, 2008).

**Overview of Contributions and Approach.** The distinctive feature of this work is the treatment of the constraints (2)-(3) in the optimization (4). In particular, gradient-based optimization as in (Paris and Le Cadre, 2002; Mihaylova et al., 2003b; Singh et al., 2007) is not suitable unless a good starting guess is chosen since the constraints impose many local minima. In addition, special differentiation (Clarke et al., 1998) is required to guarantee convergence due to the non-smooth nature of the constraints.

To overcome these issues we instead employ a methodology based on global exploration of the solution space of vehicle trajectories. This is achieved through a *random tree* of feasible trajectories. Such a tree is constructed following ideas from sampling-based motion planning (LaValle, 2006). The key property of motion planning trees relevant to this work is that the tree is guaranteed to reach asymptotically close to any reachable state in the state space as the algorithm iterates. Yet, the problem (4) is more difficult than a typical motion planning problem because the cost is based on uncertainty that depends on the whole trajectory. In essence, the problem cannot be cast as a graph or tree search typically employed in motion planning, i.e. to solve shortest path problems, because the cost function (4) is not derived from a local metric and is not additive over separate trajectory segments. Additional tools are necessary. The solution proposed in this work is to perform stochastic optimization over a solution space encoded through a dynamically adaptive trajectory tree.

The advantage of using a tree is that it provides a computationally efficient way to

encode multiple solution trajectories and to propagate probability distributions recursively. While a uniformly random tree can asymptotically reach an optimal solution this might be an infinitely slow process in practice. Therefore, as with most Monte Carlo methods (Rubenstein and Kroese, 2008) it is essential to exploit problem structure in order to speed up the search. We employ three variance reduction techniques to guide and accelerate the optimization:

1) The first, termed *biased sampling*, chooses tree nodes based on expected utility of improving the target estimate in order to focus tree exploration into more “promising” parts of the state space.

2) We then introduce a technique termed *shuffling* which randomly modifies the tree structure in an attempt to lower the optimal cost. This is achieved by disconnecting a subtree from its parent and connecting it to a different part of the tree. The tree parts to be modified are chosen probabilistically.

3) The third technique introduced in the paper, termed *randomized pruning*, removes existing nodes probabilistically according to their performance.

While biased sampling has been widely used to speed-up regular (i.e. deterministic) motion planning algorithms (Choset et al., 2005), pruning and shuffling have not been previously employed in the context of sampling-based motion planning under uncertainty. These proposed methods result in a significant computational speed-up compared to a random baseline algorithm. Yet, currently, under general regularity conditions and no additional assumptions about the structure of the HMM (1) and constraints (2)-(3), formally only asymptotic convergence rates (i.e. as the number of iterations tends to infinity) are possible. Nevertheless, there is a sound reason why the combination of these three techniques is effective. A successful optimization methodology must address the exploration-exploitation trade-off paradigm (see e.g. Powell, 2007) and do so with computational efficiency by dynamically adjusting the search space. The proposed optimization algorithm accomplishes that. In particular, the basic random tree expansion achieves exploration of the state space. The biased sampling and shuffling steps exploit information known a priori and collected during the algorithm operation to focus the search in more promising parts of the state space. Pruning is critical for maintaining a balance between the size and quality of the search space in order to achieve computational efficiency. These properties are summarized in the following table

Technique	Exploration	Exploitation	Comp. Efficiency
<i>Tree Expansion</i>	✓		
<i>Biased Sampling</i>	✓	✓	
<i>Shuffling</i>		✓	✓
<i>Pruning</i>			✓

and will be developed in detail in the paper.

**Links to Evolutionary Computing.** The resulting approach has close links to evolutionary computing. In particular, biased sampling and pruning based on an importance function correspond to *selection* from a “fitness” criteria employed in genetic algorithms. Shuffling is related to *cross-over* and *migration* used in genetic programming (Langdon and Poli, 2001) since a shuffle generates new trajectories by combining existing segments. A standard genetic algorithm could be used to perform the optimization (4) but will have difficulty managing the constraints (2) and (3) (e.g. see Michalewicz and Schoenauer, 1996). Standard techniques, i.e. penalty functions or infeasible path rejection, employed in works such as (Xiao et al., 1997; Vaidyanathan et al., 2001; Hocaoglu and Sanderson, 2001; Erinc and Carpin, 2007) depend on parametrized paths and on cost function tuning parameters. It is not clear how their performance scales as the environment becomes more cluttered. In contrast, sampling-based trees are specifically developed to handle systems with complicated dynamics and obstacle constraints. Therefore, this work employs a general motion tree to automatically encode feasible candidate paths and avoid problem-specific parameter

tuning. The stochastic optimization then amounts to dynamically adapting the tree structure towards converging to an optimal trajectory. While shuffling and pruning might seem akin to standard genetic operation, there is an important distinction—they are designed to operate over a “population” encoded as a tree of trajectories rather than as separate paths as in a standard genetic algorithm. In that sense the proposed techniques are unique and make a bridge between evolutionary algorithms and randomized motion planning methods.

## 2 An Example Scenario

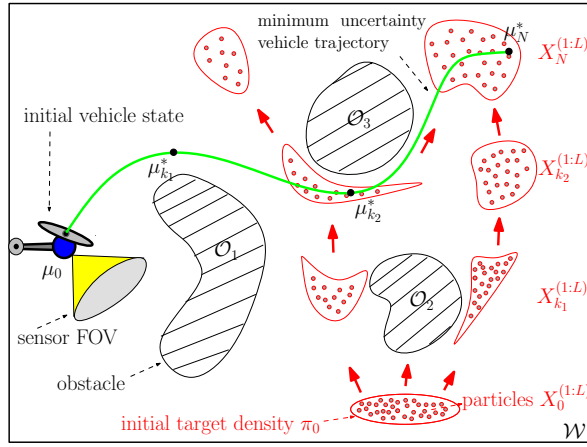


Figure 1: A scenario with a vehicle (depicted as a small helicopter) at state  $\mu_0 \in \mathcal{M}$  and a target with initial distribution  $\pi_0$  diffusing north. Both target and vehicle avoid obstacles denoted  $\mathcal{O}_i$ . The set of possible target motions is approximated by  $L$  sampled trajectories  $X_{0:N}^{(\ell)}$  for  $\ell = 1, \dots, L$ . The figure shows the sampled states (particles) at the beginning  $k = 0$ , at two intermediate times  $0 \leq k_1 \leq k_2 \leq N$ , and at the horizon  $k = N$ . We seek to find the vehicle trajectory  $\mu_{0:N}^*$  which minimizes the expected target state estimate uncertainty. The vehicle sensor typically has a small field-of-view (FOV) relative to the environment size.

Consider a scenario depicted in Fig. 1. The vehicle and target operate in a workspace (i.e. an environment) denoted by  $\mathcal{W}$ , where  $\mathcal{W} = \mathbb{R}^2$ , or  $\mathcal{W} = \mathbb{R}^3$  (Latombe, 1991). The workspace contains a number of *obstacles* denoted by  $\mathcal{O}_1, \dots, \mathcal{O}_{n_o} \subset \mathcal{W}$  with which the vehicle must not collide.

The vehicle state is defined as  $\mu = (r, v) \in \mathcal{C} \times \mathbb{R}^{n_v}$  consisting of its configuration  $r \in \mathcal{C}$  and velocity  $v \in \mathbb{R}^{n_v}$ .  $\mathcal{C}$  is the vehicle configuration space describing e.g. the position, orientation, and joint angles of the system. Assume that the vehicle occupies a region  $\mathcal{A}(r) \subset \mathcal{W}$  and that the function  $\mathbf{prox}(\mathcal{A}_1, \mathcal{A}_2)$  returns the closest Euclidean distance between two sets  $\mathcal{A}_{1,2} \subset \mathcal{W}$  and is negative if they intersect. One of the constraints defined in (3) is then to avoid obstacles, generally expressed as

$$h_c^1((r, v)_k) = \min_i \mathbf{prox}(\mathcal{A}(r_k), \mathcal{O}_i), \text{ for all } 0 \leq k \leq N. \quad (5)$$

The framework developed in the paper will be applied to two types of vehicles. The first has a simple first order model and operates in a polygonal obstacle environment (i.e.  $\dim(\mathcal{W})=2$ )—a setting suitable for measuring the algorithm performance compared to an idealized scenario. The second scenario is based on a low-flying underactuated UAV operating in a mountainous terrain in 3-D (i.e.  $\dim(\mathcal{W})=3$ ). The simpler model is presented next while the helicopter application will be developed in §7.2.

**A simple vehicle.** Consider a point mass vehicle moving in the plane. Its state space is  $\mathcal{M} = \mathbb{R}^2 \times \mathbb{R}^2$  with state  $\mu = (r, v)$  consisting of the position  $r := (r_x, r_y) \in \mathbb{R}^2$  and velocity  $v := (v_x, v_y) \in \mathbb{R}^2$ . It evolves according to the simple dynamics

$$r_{k+1} = r_k + \tau v_k, \quad (6)$$

which is encoded by the function  $h_d$  defined in (2). The constant  $\tau$  is the *time-step*, i.e. the sampling period, measured in seconds. The velocity  $v_k$  can be directly controlled but is bounded  $\|v_k\| < v_{\max}$ . For instance, in the scenario (Fig. 1) a bound of  $v_{\max} = 8m/s$  is chosen to create a problem that can be solved optimally only by a particular type of trajectory known in advance for a time horizon of 30 seconds.

**Target dynamics.** The target is modeled as a point mass on the ground with position  $r = (r_x, r_y) \in \mathbb{R}^2$ , velocity  $v = (v_x, v_y) \in \mathbb{R}^2$  forming the state  $x = (r, v)$  with  $\mathcal{X} = \mathbb{R}^2 \times \mathbb{R}^2$ .

The target dynamics is governed by a general control law including a proportional term, such as arising from a goal attraction, a damping term in order to constrain the target speed, and obstacle avoidance forcing. In addition, there is white noise acceleration component  $\Omega$  with standard deviations  $\sigma_x$  and  $\sigma_y$ . The model is

$$\begin{aligned} r_{k+1} &= r_k + \tau v_k, \\ v_{k+1} &= v_k + \tau \left( \omega_k + K_p(r_g - r_k) - K_d v_k + \begin{bmatrix} 0 & -k_o/d_k \\ k_o/d_k & 0 \end{bmatrix} v_k \right), \\ \omega_k &\sim \text{Normal}(0, \text{diag}(\sigma_x^2, \sigma_y^2)), \end{aligned} \quad (7)$$

where  $\tau$  is the time-step;  $K_p > 0$  and  $K_d > 0$  are potential and dissipative matrices, respectively;  $k_o > 0$  is an obstacle steering scalar gain;  $r_g \in \mathbb{R}^2$  is a constant goal location;  $d_k = \|g_k\|$  with  $g_k := r_o - r_k$ , where  $r_o$  is the closest point on the obstacle set. The model (7) corresponds to the function  $f$  defined in (1a).

Such a model is chosen in order to add realism to the target motion as if it were executing an actual navigation task. The amount of knowledge of this task can be tuned using the provided parameters. For instance, in the numerical scenarios studied in the paper we use

$$\begin{aligned} K_p &= \text{diag}(0, 0.03), \quad K_d = \text{diag}(0, 0.6), \quad r_g = (0, 200), \\ k_o &= 50, \text{ if } |\beta_k| < \pi/2, \text{ else } k_o = 0, \end{aligned}$$

where  $\beta_k$  is angle between  $v_k$  and  $g_k$ . Assuming the target starts at the origin, the dynamics would drive it North, avoiding obstacles if too close, diffusing in uncertain directions to East or West, and finally ending up and staying around the boundary line  $r_y = 200$ .

**Sensor Model.** The vehicle is equipped with sensors which provide relative range and bearing to target, hence  $\mathcal{Y} = \mathbb{R}^2$ . The target is observed only if its line of sight is not obstructed by obstacles and if it falls within the *sensing distance*  $d_s$  of the vehicle. This is formally expressed through the target *visibility area*  $\mathcal{V}(r') \subset \mathcal{W}$  for given vehicle position  $r' = (r'_x, r'_y) \in \mathbb{R}^2$  defined by

$$\mathcal{V}(r') := \{r \in \mathcal{W} \mid \|r - r'\| < d_s \text{ and } h_c^1((r' + \alpha(r - r'), \cdot)) \geq 0, \forall \alpha \in [0, 1]\}.$$

In order to define the sensor function (1b) for a target with position  $r = (r_x, r_y)$  first define the *perfect* sensor function

$$g^*((r, v); (r', v')) = \begin{bmatrix} \|r - r'\| \\ \arctan((r_y - r'_y), (r_x - r'_x)) \end{bmatrix}, \quad (8)$$

This function is only valid if the target reading originated from its visibility region. In addition, there is a small probability  $P_f \in [0, 1]$  of a false reading uniformly distributed over the visibility region. The actual sensor function is then given by

$$g((r, v); (r', v'), V) = \begin{cases} g^*((r, v); (r', v')) + \frac{\|r - r'\|}{d_s} V, & r \in \mathcal{V}(r') \\ \emptyset, & r \notin \mathcal{V}(r') \end{cases} \text{ if } u_f \geq P_f \quad (9)$$

$$g^*((r^f, v); (r', v')), \quad r^f \sim \mathcal{U}(\mathcal{V}(r')) \text{ if } u_f < P_f.$$

with noise  $V := (V_d, V_b) \sim \text{Normal}(0, \text{diag}(\sigma_d^2, \sigma_b^2))$  where  $\sigma_d$  and  $\sigma_b$  define the range and bearing standard deviations, respectively, and  $u_f$  is a uniform sample from  $[0, 1]$ .

**Cost function.** Finally, the vehicle is interested in minimizing the uncertainty in the target position estimate. This is encoded by simply setting

$$\varphi(x) = (r_x, r_y) \in \mathbb{R}^2$$

when performing the optimization (4).

### 3 Problem Formulation

An alternative way to express the HMM (1) is through the known densities

$$X_0 \sim \pi_0, \quad (10a)$$

$$X_k \sim p(\cdot | X_{k-1}), \quad k > 0 \quad (10b)$$

$$Y_k \sim q(\cdot | X_k; \mu_k), \quad k > 0 \quad (10c)$$

where  $\pi_0$  is the initial distribution. Note that the expectation operator  $\mathbb{E}[\cdot]$  used throughout the paper is applied with respect to these densities, unless noted otherwise. The filtering density employed in the computation of the cost (4) is then expressed recursively (e.g. Robert and Casella, 2004) according to

$$\pi_k(x | y_{1:k}; \mu_{1:k}) = \frac{q(y_k | x; \mu_k) \int p(x | x') \pi_{k-1}(x' | y_{1:k-1}; \mu_{1:k-1}) dx'}{\int q(y_k | x; \mu_k) \int p(x | x') \pi_{k-1}(x' | y_{1:k-1}; \mu_{1:k-1}) dx' dx}. \quad (11)$$

In addition, the tree optimization algorithm will require the definition of the *prediction density* at time  $k + i$ , for  $i > 0$ , after receiving measurements only during the first  $k$  epochs. It is denoted  $\pi_{k+i|k}$  and defined by

$$\pi_{k+i|k}(x | y_{1:k}; \mu_{1:k}) = \int p(x | x') \pi_{k+i-1|k}(x' | y_{1:k}; \mu_{1:k}) dx', \quad (12)$$

with  $\pi_{k|k} \equiv \pi_k$ . The estimate of  $\varphi(X_N)$  after collecting a sequence of measurements  $y_{1:k}$  obtained from a vehicle trajectory  $\mu_{1:k}$  is denoted  $\Phi_{N|k} : \mathcal{Y}^k \times \mathcal{M}^k \rightarrow \mathbb{R}^{n'_x}$  and defined by

$$\Phi_{N|k}(y_{1:k}; \mu_{1:k}) := \mathbb{E}[\varphi(X_N) | y_{1:k}; \mu_{1:k}] = \int \varphi(x_N) \pi_{N|k}(x | y_{1:k}; \mu_{1:k}) dx. \quad (13)$$

The objective function in (4) or, equivalently, the expected *uncertainty cost* at time  $N$  given a vehicle trajectory  $\mu_{0:k}$ , for  $k \leq N$  is denoted by  $J_{N|k} : \mathcal{M}^k \rightarrow \mathbb{R}$  and defined according to

$$J_{N|k}(\mu_{1:k}) = \mathbb{E}[\|\varphi(X_N) - \Phi_{N|k}(Y_{1:k}; \mu_{1:k})\|^2] \\ = \int \|\varphi(x_N) - \Phi_{N|k}(y_{1:k}; \mu_{1:k})\|^2 p(x_{0:N}, y_{1:k} | \mu_{1:k}) dx_{0:N} dy_{1:k}. \quad (14)$$

The expectation over states and measurements in (14) is taken with respect to the density  $p(x_{0:N}, y_{1:k} | \mu_{1:k})$  which, for Markov models in the form (10), can be decomposed (see e.g. (Robert and Casella, 2004)) as

$$p(x_{0:N}, y_{1:k} | \mu_{1:k}) = \pi_0(x_0) \prod_{i=1}^N p(x_i | x_{i-1}) \prod_{i=1}^k q(y_i | x_i; \mu_i). \quad (15)$$

The cost of a complete trajectory  $\mu_{1:N}$  is denoted for brevity by  $J := J_{N|N}$ . The goal (4) is then expressed in short as

$$\mu_{1:N}^* = \arg \min_{\mu_{1:N}} J(\mu_{1:N}), \quad (16)$$

subject to the dynamics and constraints.

## 4 Sampling-based Approximation

The filtering densities (11) and (12) generally cannot be computed in closed form since they are based on nonlinear/non-Gaussian models. Therefore, following (Singh et al., 2007), we employ particle-based approximation using  $L$  delta distributions, placed at state *samples*  $X_k^{(j)} \in \mathcal{X}$  with positive weight functions  $w_k^{(j)} : \mathcal{Y}^k \times \mathcal{M}^k \rightarrow \mathbb{R}_+$ , i.e.

$$\pi_k(x | y_{1:k}; \mu_{1:k}) \approx \hat{\pi}_k(x | y_{1:k}; \mu_{1:k}) := \sum_{j=1}^L w_k^{(j)}(y_{1:k}; \mu_{1:k}) \delta_{X_k^{(j)}}(x), \quad (17)$$

where  $\delta_y$  denotes the Dirac delta mass at point  $y$ .

A simple way to construct such a representation is to sample  $L$  independent trajectory realizations  $\{X_{0:k}^{(j)}\}_{j=1}^L$  using the prior (10a) and target motion model (10b) and to compute the weights, for given measurements  $y_{1:k}$  obtained at vehicle states  $\mu_{1:k}$ , according to

$$\bar{w}_k^{(j)}(y_{1:k}; \mu_{1:k}) := \prod_{i=1}^k q(y_i | X_i^{(j)}; \mu_i), \quad (18)$$

$$w_k^{(j)} = \frac{\bar{w}_k^{(j)}}{\sum_{\ell=1}^L \bar{w}_k^{(\ell)}}, \quad (19)$$

so that the weights are normalized, i.e.  $\sum_{j=1}^L w_k^{(j)} = 1$ . This is equivalent to a sequential importance sampling (SIS) scheme with importance distribution  $\pi_0(x_0) \prod_{i=1}^k p(x_i | x_{i-1})$ . Note that while more sophisticated sampling methods have been developed, e.g. that additionally account for measurements to reduce variance (Doucet et al., 2001; Robert and Casella, 2004), this work follows the basic choice for simplicity. Fig. 1 depicts a subset of possible evolutions of such particles in the helicopter search scenario.

With this representation it is straightforward to show that (12) is approximated simply according to

$$\pi_{k+i|k}(x | y_{1:k}; \mu_{1:k}) \approx \hat{\pi}_{k+i|k}(x | y_{1:k}; \mu_{1:k}) := \sum_{j=1}^L w_k^{(j)}(y_{1:k}; \mu_{1:k}) \delta_{X_{k+i}^{(j)}}(x). \quad (20)$$

The estimate (13) is then approximated by

$$\Phi_{N|k}(y_{1:k}; \mu_{1:k}) \approx \hat{\Phi}_{N|k}(y_{1:k}; \mu_{1:k}) := \sum_{j=1}^L \varphi(X_N^{(j)}) \hat{\pi}_{N|k}(X_N^{(j)} | y_{1:k}; \mu_{1:k}). \quad (21)$$

Note that updating the cost along a vehicle trajectory has computational complexity  $O(L^2)$  per time step. Yet, due to particle independence the computation can be parallelized using special hardware up to a factor of  $O(L)$  and sped up significantly.



As the time  $N$  increases the approximation (21) degrades since the probability mass becomes concentrated in a decreasing number of particles (Robert and Casella, 2004). A standard remedy is to include a resampling step (Doucet et al., 2001) to redistribute the samples equalizing the weights. While it is possible to perform sequential importance resampling (SIR) in the proposed framework it is avoided for computational reasons specific to the tree structure employed for uncertainty propagation. The drawback is that the method is limited to small time horizons, e.g.  $N < 30$ . The distinct advantage though is that the simpler SIS scheme permits a computationally efficient update of the density (17) and estimate (21) during optimization. The idea (described in detail in the following sections) is that SIS can be implemented as a *simple and fast* parallel weights rescaling in a dynamically changing tree of vehicle trajectories that explores the solution space.

Finally, the error  $J_{N|k}$  is approximated through importance sampling of the integrand in (14), i.e. by drawing  $(X_{0:N}^{(\ell)}, Y_{1:k}^{(\ell)})$  from  $p(x_{0:N}, y_{1:k} | \mu_{1:k})$ . It is natural to use the i.i.d. state particles  $X_{0:N}^{(\ell)}$  already sampled for the approximation of the density (17). Measurement sequences  $Y_{1:k}^{(\ell)}$  are then sampled by drawing  $Y_i^{(\ell)} \sim q(\cdot | X_i^{(\ell)}; \mu_i)$  for all  $i = 1, \dots, k$ . As long as the densities (10) can be directly sampled from, which is valid for common models used in robotics (e.g. Thrun et al., 2005), then the approximation simplifies to the Monte Carlo or the stochastic counterpart, i.e.

$$J_{N|k}(\mu_{1:k}) \approx \hat{J}_{N|k}(\mu_{1:k}) := \frac{1}{L} \sum_{\ell=1}^L \|\varphi(X_N^{(\ell)}) - \hat{\Phi}_{N|k}(Y_{1:k}^{(\ell)}; \mu_{1:k})\|^2, \quad (22)$$

with  $\hat{J} := \hat{J}_{N|N}$  denoting the approximate cost of a whole trajectory  $\mu_{1:N}$ . The global optimization algorithms developed in the paper will be based on the approximate estimate (21) and cost (22), i.e. it will solve

$$\hat{\mu}_{0:N}^* = \arg \min_{\mu_{0:N}} \hat{J}(\mu_{0:N}). \quad (23)$$

In this sense only an approximate solution will be obtained. Yet, by the law of large numbers (Del Moral, 2004)  $\hat{\mu}_{0:N}^*$  will approach the true solution  $\mu_{0:N}^*$  by increasing the number of simulations  $L$ .

## 5 Random Tree Optimization

The nature of the constraints (2) and (3) renders gradient-based methods unsuitable for solving (23). An alternative is to discretize the vehicle state space  $\mathcal{M}$ , e.g. using a grid and generate candidate paths by transitioning between adjacent cells. Such an approach is generally limited to few dimensions, e.g.  $\dim(\mathcal{M}) \leq 3$  and to systems with very simple dynamics, e.g. an unconstrained point mass in the plane. This is due to the exponential (both in state dimension and trajectory epochs) size of the search space, also known as the *curse of dimensionality*.

In this paper we also employ tree-based search but unlike in standard discrete search, the nodes of the tree are *sampled* from the original continuous space  $\mathcal{M}$  and the edges correspond to trajectories satisfying any given dynamics (2) and general constraints (3). Our approach is based on a recent methodology under active development in the robotics community known as *sampling-based motion planning* which includes the *rapidly-exploring random tree* (RRT) (LaValle, 2006) and the *probabilistic roadmap* (PRM) (Choset et al., 2005). Unlike these motion planning algorithms though the trees employed in this work are not expanded based on a “distance” metric between nodes. Instead, the connections are made probabilistically, nodes and edges can be added, swapped, or deleted during the algorithm operation. This section considers the basic tree expansion that explores the state space, while §6 introduces the variance reduction techniques that complete the overall approach.

Variable	Type	Element Description
$k^b$	integer $\geq 0$	time epoch index
$\mu^b$	$\dim(\mathcal{M}) \times 1$ vector	vehicle state at node
$W^b$	$L \times L$ matrix	weights $W_{\ell j}^b := w_{k^b}^{(j)}(Y_{1:k^b}^{(\ell)}; \mu^{0 \rightarrow b})$ , for $\ell, j = 1 \dots L$
$\hat{J}^b$	scalar $> 0$	uncertainty cost $\hat{J}^b := J_{N k^b}(\mu^{0 \rightarrow b})$
$\rho^b$	integer $\geq 0$	parent node index

Figure 2: Description of the elements of a node  $\eta^b = (k^b, \mu^b, W^b, \hat{J}^b, \rho^b) \in \mathcal{T}$ .

## 5.1 Tree Expansion

The set of nodes is denoted by  $\mathcal{N} := \mathbb{N} \times \mathcal{M} \times \mathbb{R}^{L \times L} \times \mathbb{R}_+ \times \mathbb{N}$ . Each *node* is defined by the tuple

$$\eta = (k, \mu, W, \hat{J}, \rho) \in \mathcal{N},$$

consisting of: the epoch index  $0 \leq k \leq N$ ; vehicle state  $\mu$ ; particle weights matrix  $W$  providing a convenient way to compute the density  $\hat{\pi}$ ; target state estimate uncertainty cost  $\hat{J} \geq 0$ ; tree parent index  $\rho$ . Nodes and their sub-elements are indexed by superscript, i.e.  $\eta^a$  has state  $\mu^a$  and its parent node is  $\eta^{\rho^a}$ . The *root* of the tree that contains the starting vehicle state is denoted  $\eta^0 = (0, \mu_0, W^0, \hat{J}^0, \cdot)$ , where the matrix elements  $W_{\ell j}^0 = 1/L$  for all  $\ell, j = 1, \dots, L$ . A *trajectory* between two nodes  $\eta^a$  and  $\eta^b$  is denoted  $\mu^{a \rightarrow b}$  and a state at time  $k$  along this trajectory is denoted  $\mu_k^{a \rightarrow b}$  where  $k^a \leq k \leq k^b$ .

A tree  $\mathcal{T} \subset \mathcal{N}$  is a set of nodes connected by feasible trajectories. The tree structure guarantees that there is a unique trajectory leading from the root to each node  $\eta^a \in \mathcal{N}$  which is denoted  $\mu^{0 \rightarrow a}$ . Overview of the elements comprising each node is given in Fig. 2. Their exact computation is detailed next.

A tree is constructed by assuming that a local controller is available (LaValle, 2006) that attempts to drive the vehicle between two given nodes  $\eta^a$  and  $\eta^b$ . For instance, if the states  $\mu^a$  and  $\mu^b$  were close enough and no obstacles between them were present then a trajectory  $\mu^{a \rightarrow b}$  is produced, otherwise the connection fails. Such a controller is abstractly represented by the function **Connect**, i.e.

$$\mathbf{Connect}(\eta^a, \eta^b) \Rightarrow \begin{cases} \mu^{a \rightarrow b}, & \text{if path found} \\ \emptyset, & \text{otherwise.} \end{cases} \quad (24)$$

The tree is constructed by sampling and connecting nodes. Assume that a function **Sample** is available which returns a new node, i.e.

$$\eta^b = \mathbf{Sample}(). \quad (25)$$

The default choice is to sample (state,time) pairs  $(\mu, k)$  uniformly from  $\mathcal{M} \times \{1, \dots, N\}$  and discard samples that violate the constraints (3), e.g. that lie inside obstacles. Next define the set  $\mathcal{T}^{\rightarrow b} \subset \mathcal{T}$  of all existing tree nodes for which a feasible trajectory to the newly sampled node can be found, i.e.

$$\mathcal{T}^{\rightarrow b} = \{\eta \in \mathcal{T} \mid \mathbf{Connect}(\eta, \eta^b) \neq \emptyset\}.$$

One of these nodes denoted  $\eta^a \in \mathcal{T}^{\rightarrow b}$  is selected uniformly at random to become the parent of  $\eta^b$  linked with trajectory  $\mu^{a \rightarrow b}$ , i.e.  $\rho^b = a$ . Fig. 3 illustrates the construction.

After a new node  $\eta^b$  is added to the tree, the target filtering density  $\hat{\pi}$  (20) is propagated along the newly added trajectory segment  $\mu^{a \rightarrow b}$  for all sampled target paths  $X_{k^a:k^b}^{(\ell)}$  by simulating measurements

$$Y_k^{(\ell)} \sim q(\cdot | X_k^{(\ell)}; \mu_k^{a \rightarrow b}), \quad \text{for } k = k^a, \dots, k^b, \quad (26)$$

for all  $\ell = 1 \dots L$ . A row in the matrix  $W^b$ , i.e.  $W_\ell^b$  for any  $1 \leq \ell \leq L$  corresponds to the resulting weights for each measurement sequence, i.e.  $W_{\ell j}^b := w_{k^b}^{(j)}(Y_{k^b}^{(\ell)}; \mu^{0 \rightarrow b})$ ,

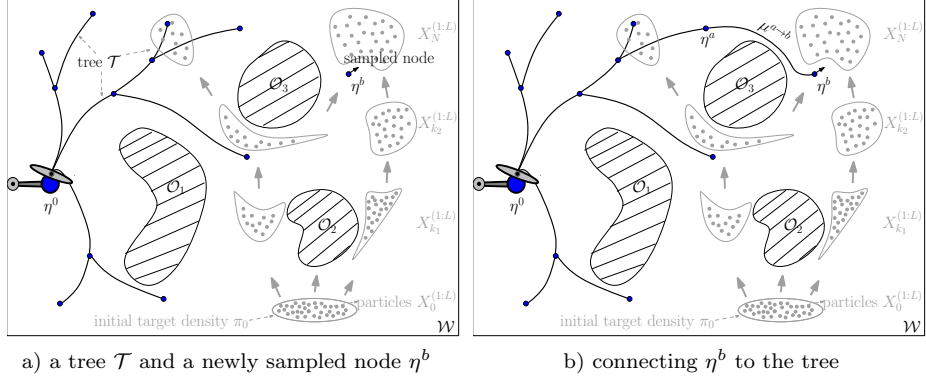


Figure 3: A tree expansion step implemented by **Expand** (§5.1): a) a node  $\eta^b$  is sampled; b) it is then connected to a randomly chosen node  $\eta^a \in \mathcal{T}$

where  $w_{k^b}^{(j)}$  is defined in (18). The weights are computed *incrementally* using the parent weights  $W_{\ell_j}^a$  through

$$\bar{W}_{\ell_j}^b = W_{\ell_j}^a \cdot U_{\ell_j}, \text{ where } U_{\ell_j} := \prod_{k=k^a}^{k^b} q(Y_k^{(\ell)} | X_k^{(j)}; \mu_k^{a \rightarrow b}), \quad (27a)$$

$$W_{\ell_j}^b = \frac{\bar{W}_{\ell_j}^b}{\sum_{j=1}^L \bar{W}_{\ell_j}^b}. \quad (27b)$$

The error (22) of the complete trajectory  $\mu^{0 \rightarrow b}$ , denoted by  $\hat{J}^b$ , becomes

$$\hat{J}^b := \hat{J}_{N|k^b}(\mu^{0 \rightarrow b}) = \sum_{\ell=1}^L \|\varphi(X_N^{(\ell)}) - \sum_{j=1}^L W_{\ell_j}^b \varphi(X_N^{(j)})\|^2. \quad (28)$$

Note again that  $\hat{J}^b$  represents the uncertainty measure at the end of the time horizon  $N$  but only based on measurements collected along  $\mu^{0 \rightarrow b}$ , i.e. up to time  $k^b$ . If  $\hat{J}^b < \hat{J}^*$ , where  $\hat{J}^b$  is computed using (28) and  $\hat{J}^*$  is the current best cost, then the current best node is reset, i.e.  $\eta^* = \eta^b$ . The updated optimal vehicle trajectory can be backtracked from  $\eta^b$  to the root  $\eta^0$ .

Let  $s \sim \mathcal{U}(\mathcal{S})$  denote uniform sampling of an element  $s$  from a finite set  $\mathcal{S}$ . The complete tree expansion algorithm can now be summarized as

**Expand**

1.  $\eta^b = \text{Sample}()$
2.  $\eta^a \sim \mathcal{U}(\mathcal{T}^{\rightarrow b})$
3.  $\mu^{a \rightarrow b} = \text{Connect}(\eta^a, \eta^b)$
4.  $\mathcal{T} = \mathcal{T} \cup \{\eta^b\}$ ;  $\rho^b = a$
5. Compute  $W_{\ell_j}^b$  and  $\hat{J}^b$  using (27) and (28)
6. if  $\hat{J}^b < \hat{J}^*$  then  $\eta^* = \eta^b$ .

The expansion is repeated  $n - 1$  times in order to produce a tree with  $n$  nodes. Initially, the tree contains only the root, i.e.  $\mathcal{T} = \{\eta^0\}$ , and  $\eta^* = \eta^0$ .

**Computational Saving.** It is important to stress that the computation (28) is accomplished through an *incremental propagation* of the filtering density weights along the newly added trajectory  $\mu^{a \rightarrow b}$  from parent  $\eta^a$  to child node  $\eta^b$  rather than the complete trajectory  $\mu^{0 \rightarrow b}$ . This signifies the advantage of using a tree rather than a naive enumeration of vehicle trajectories in order to explore the solution space  $\mathcal{M}^k$ . For instance, assume that the tree were a complete binary tree with  $n$  nodes and, hence, with depth  $d = \log(n+1) - 1$ . Then it encodes  $n/2$  different trajectories since each leaf

can be backtracked to generate a unique trajectory  $\mu_{0:N}$ . If each edge lasts on average of  $N/d$  time epochs then the density computation (27) must be performed  $\frac{n}{d}N$  times for the tree compared to  $\frac{n}{2}N$  times if the  $n/2$  trajectories were enumerated. In other words, the tree provides a  $O(\log(n))$  savings factor on average.

## 5.2 Example: simple vehicle

Consider the simple vehicle with dynamics (6). Since there is no bound on accelerations a trajectory between two nodes is simply a line segment with constant velocity. The function **Connect** introduced in §5.1 takes the form

```

Connect( $\eta^a, \eta^b$ )
1.  if  $k^b = \infty$ ,
2.     $k^b = k^a + \left\lceil \frac{\|r^b - r^a\|}{\tau v_{max}} \right\rceil$ 
3.     $v = \frac{r^b - r^a}{\tau(k^b - k^a)}$ 
4.    for  $k = k^a : k^b$ 
5.       $r_k = r^a + \frac{k - k^a}{k^b - k^a} (r^b - r^a)$ ;  $\mu_k^{a \rightarrow b} = (r_k, v)$ 
6.      if  $h_c^1(\mu_k^{a \rightarrow b}) < 0$  return  $\emptyset$ 
7.    return  $\mu^{a \rightarrow b}$ 

```

It begins by checking whether the trajectory to be computed must have fixed final time  $k^b$ . When the final time epoch  $k^b$  is set to  $\infty$  (line 1) then any  $k^b$  such that  $k^a < k^b \leq N$  is allowed. This occurs when  $\eta^b$  is a uniform sample<sup>2</sup> in which case the trajectory  $\mu^{a \rightarrow b}$  is generated using maximum permitted velocity  $v_{max}$  and  $k^b$  is set to the resulting time (line 2). The constant velocity along the trajectory is computed in line 3. The points along the trajectory are then linearly interpolated (line 5). If the trajectory intersects obstacles then the connection fails (line 6).

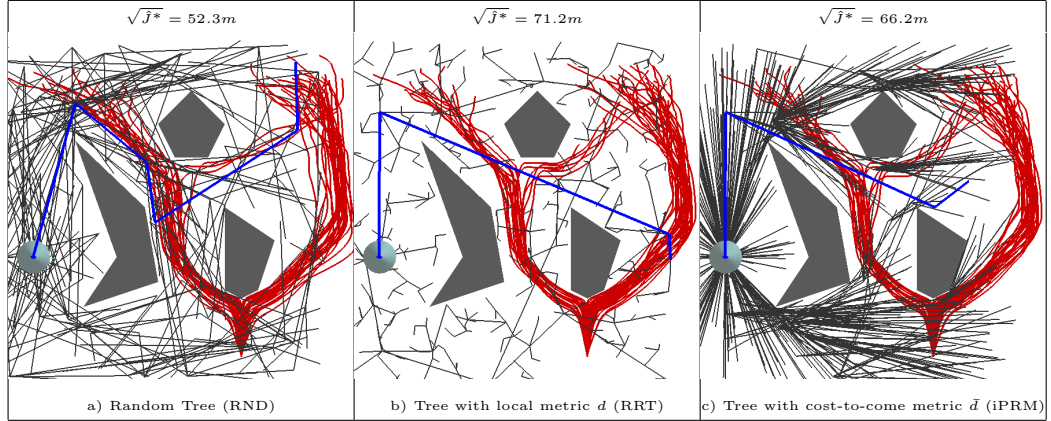


Figure 4: Three types of search trees used to explore the vehicle trajectory space corresponding to the scenario in §2. The vehicle has simple dynamics and a circular sensing radius shown as a disk at its starting state. A subset of the target paths  $X_{0:N}^{(1:L)}$  are shown diffusing from the bottom right to the top of the environment. The trees are: a) a random tree (RND) constructed using **Expand** (§5.1); b) a rapidly-exploring random tree (RRT) using nearest neighbor metric  $d$  (29); c) an incremental tree-based probabilistic roadmap (iPRM) expanded based on cost-to-come distance  $\tilde{d}$  (30). Each tree has 500 nodes. While the optimal (i.e. with minimum target position variance) trajectories of all trees are quite different (shown as thicker lines), they all yield very similar costs  $\hat{J}^*$ . It is evident that these solutions are of *poor quality* since the square root of the variance is large (i.e.  $> 52.3$  meters) relative to the environment size ( $200 \times 200$  meters.).

The optimal estimation algorithm is tested in a polygonal obstacle environment

<sup>2</sup>a uniform sample is sampled uniformly over the state space; additional sampling choices will be given in §6.1.

mimicking the scenario in §2. A tree built after calling **Expand** 500 times is shown on Fig. 4(a). It takes a few milliseconds of computation to generate such a tree. In practice, a tree will contain tens of thousands of vertices. Fig. 5 shows a few frames of the resulting motion along an optimal trajectory obtained by a denser tree with 10000 nodes which took 5 seconds to compute. More detailed computational studies are performed in §7.

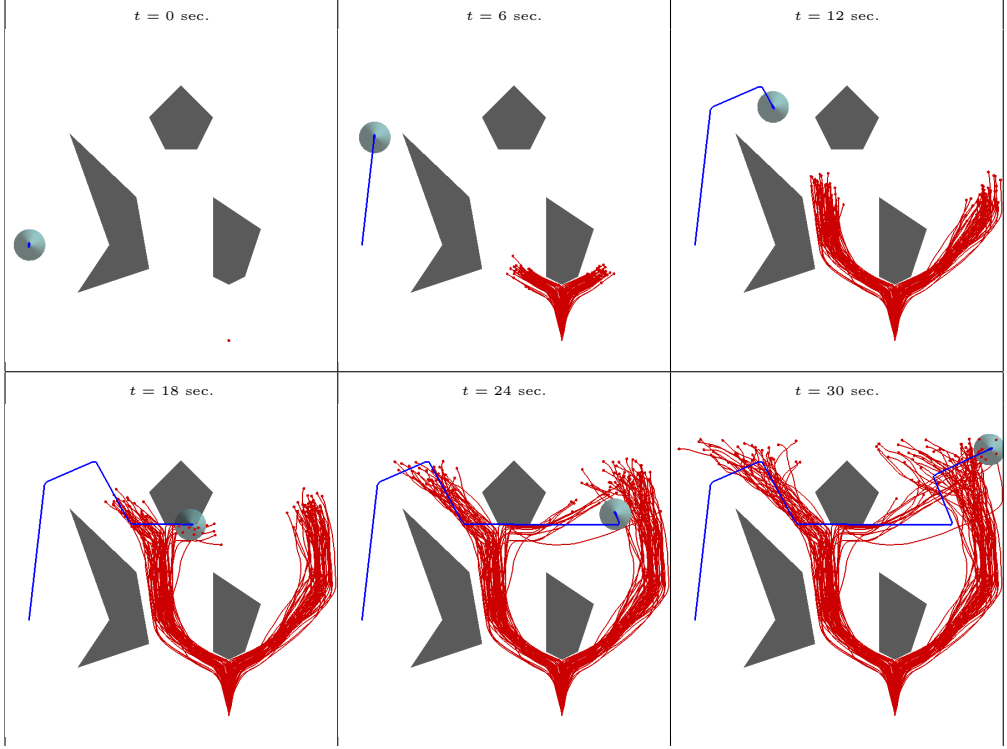


Figure 5: The optimal vehicle path computed using algorithm **Expand** (§5.1) with  $L = 100$  particles, time horizon  $T = 30$  s., and time-step  $\tau = 250$  ms. The consecutive frames show the evolution of the sampled target trajectories  $X_{0:N}^{(1:L)}$  and the vehicle trajectory  $\mu_{0:N}^*$ . The computed cost is  $\sqrt{\hat{J}^*} = 43.9m$ .

### 5.3 Other Expansion Methods

There are alternative methods to construct an exploration tree. For instance, instead of connecting nodes at random, a newly sampled node can be connected to an existing node in the tree based on some deterministic criteria. Classical planning trees such as an RRT employ *nearest* neighbor connections. Nearest should be understood with respect to a predefined pseudo<sup>3</sup> -distance metric  $d : \mathcal{N} \times \mathcal{N} \rightarrow \mathbb{R}$ . Typical metrics include the Euclidean distance  $d(\eta^a, \eta^b) = \|\mu^b - \mu^a\|$  (assuming  $\mathcal{M}$  is a vector space) or the time of travel between nodes

$$d(\eta^a, \eta^b) = k^b - k^a. \quad (29)$$

Using such a local cost  $d$  has the advantage of quickly exploring the state space. It also has a major drawback: the cumulative cost of a path (e.g. its length or total time) can significantly deviate from an existing shortest path. More intuitively, even

<sup>3</sup> $d$  is not required to be a true distance metric, i.e. does not need to be symmetric or necessary satisfy the triangle inequality (LaValle, 2006)

though the tree would reach close to all states in the domain, most of the paths will be jagged and circuitous. For instance, Fig. 4(b) shows such a tree computed for a scenario mimicking the setup on Fig. 1.

This can be remedied by considering the cost-to-come  $c$  defined by

$$c^b = c^a + d(\eta^a, \eta^b),$$

with the cost of the root  $c^0 = 0$ . Nearest neighbor is then chosen according to a modified distance  $\bar{d}$  defined by

$$\bar{d}(\eta^a, \eta^b) = c^a + d(\eta^a, \eta^b). \quad (30)$$

A tree built based on  $\bar{d}$  will contain “straighter” trajectories (Fig. 4(c)) which take minimum time to reach the reachable points in the state space. Such a tree is termed *incremental probabilistic roadmap* (iPRM) to distinguish it from the standard RRT. These issues are discussed in (Frazzoli et al., 2002) who propose a general motion planning algorithm based on a combined metric of  $d$  and  $\bar{d}$ .

More formally, the nearest existing node  $\eta^a \in \mathcal{T}$  to a sampled node  $\eta^b$  is given by

$$\eta^a = \arg \min_{\eta \in \mathcal{T} \rightarrow b} d(\eta, \eta^b). \quad (31)$$

Such nearest neighbor expansion is achieved by replacing (line 2) in routine **Expand** with (31).

While the RRT and iPRM are standard choices for motion planning problems, as we will show they are not a good option when optimizing uncertainty-based cost functions, i.e. solving (4). The issue is that the cost  $J$  of a path cannot be expressed as a summation over the costs of its individual edges. In that respect, the random algorithm RND turns out to be more effective.

## 5.4 Probabilistic Completeness

Under certain assumptions a random tree can reach asymptotically close to any state that is reachable. Recall the following lemma adapted to the settings of this work:

**Lemma 5.1.** (see e.g. Ladd and Kavraki, 2004) *After selecting  $n$  nodes, the probability of failing to find a path from the root  $\eta^0$  to a uniformly sampled node  $\eta^a \in \mathcal{N}$  reduces exponentially in  $n$ . More formally,*

$$P(\bar{\mathcal{A}}^{\mu^{0 \rightarrow a}}) < \ell(1 - c)^n,$$

for some constants  $\ell, c \in (0, 1)$ .

The main assumption in Lemma (5.1) is that each node can be connected to a sufficiently large number of other nodes using the **Connect** routine. The collection of these nodes is called the *reachable set* and its size depends on the constraints (2) and (3), e.g. it shrinks if a vehicle is slow or if the number of obstacles in the environment increases. The intuition behind Lemma (5.1) then is that, as long as every node has large enough reachable space (the volume of which is proportional to  $c$ ) and, under the assumption that the path has a finite length (related to the constant  $\ell$ ), then adding more nodes would exponentially increase the probability of finding the path. The precise technical conditions which render Lemma (5.1) applicable to the scenario of this paper are developed in (Hsu et al., 2002; Frazzoli et al., 2002).

## 6 Variance Reduction Techniques

The advantage of constructing the exploration tree described in §5 is that it asymptotically reaches arbitrary close to any state in the state space  $\mathcal{M}$  assuming that it is bounded. To guarantee that formally (as described in §5.4) the tree is constructed using

uniform node sampling and random connection of new nodes. Note that the expansion is agnostic to the uncertainty cost  $\hat{J}$  along trajectories that we actually seek to minimize. This is problematic because it might take infinitely long to explore a reasonable fraction of trajectories with low  $\hat{J}$ . Therefore, in the spirit of Monte Carlo optimization based on importance sampling (Rubenstein and Kroese, 2008) as well as genetic computation, we propose three techniques which retain the probabilistic completeness of random trees but at the same time drastically speedup the optimization.

## 6.1 Utility Sampling of Nodes

The difficulty of optimization in a complicated high-dimensional landscape can in practice be alleviated by incorporating problem-specific knowledge. For instance, the set of nodes considered during randomized motion planning can be chosen in a biased way, e.g. proportional to some *utility* function known to reduce the trajectory cost (see e.g. Burns and Brock, 2005).

This paper employs a similar approach dictated by the fact that an optimal vehicle trajectory  $\mu_{0:N}^*$ , i.e. with lowest uncertainty cost  $J$ , is likely to pass close to states with high observation likelihood. Thus, a sample  $\mu_k^{\text{sample}}$  is chosen so that

$$\mu_k^{\text{sample}} = \arg \max_{\mu} q(Y_k^\ell | X_k^\ell; \mu), \quad (32)$$

where  $(X_k^{(\ell)}, Y_k^{(\ell)})$  is a single particle selected by sampling  $\ell$  uniformly from  $\{1, \dots, L\}$ . The optimal state  $\mu$  in (32) is usually straightforward to compute. For instance, the optimal vehicle position in the example scenario from §2 will coincide (on average) with the target position at  $X_k^\ell$  so according to (32) one can simply set  $\mu_k^{\text{sample}} = X_k^\ell$ .

It is also possible to sample  $\mu$  by minimizing the joint likelihood over all particles. Since this results in a complex multimodal optimization subject to sensor visibility constraints (such as (9)) we choose to use the simplest form (32) as it does not add extra complexity to the overall algorithm.

The function **Sample** introduced in §5.1 is specified as follows. It samples a state  $\mu$  in two ways: 1) based on the utility (32); 2) uniformly in the space  $\mathcal{M}$ . It selects the former with probability  $P_U$ , otherwise it selects the latter at every tree expansion. The routine is summarized as

### Sample

1. With probability  $P_U$ ,
2.  $k \sim \mathcal{U}(\{0, \dots, N\})$ ;  $\ell \sim \mathcal{U}(\{1, \dots, L\})$ ;
3.  $\mu = \arg \max_{\mu'} q(Y_k^\ell | X_k^\ell; \mu')$ , where  $Y_k^\ell \sim q(\cdot | X_k^\ell; \mu')$
4. otherwise
5.  $k = \infty$
6.  $\mu \sim \mathcal{U}(\mathcal{M})$
7. repeat Sample if  $h_c(\mu) < 0$
8. return  $\eta = (k, \mu, \dots)$

A sampled node is accepted as long as it satisfies the constraints (line 7). A utility node is sampled by choosing its time epoch  $k$  (line 2) while a uniform node has time  $k = \infty$  (line 5). This is related to the way the function **Connect**( $\eta^a, \eta^b$ ) links two nodes  $\eta^a$  and  $\eta^b$ . Whenever  $k^b = \infty$ , **Connect** is allowed to produce a trajectory  $\mu^{a \rightarrow b}$  with any final time  $k^b \leq N$  that it chooses. In such case **Connect** is typically designed to compute a time-optimal trajectory so that  $k^b$  is minimized. In contrast, when  $k^b$  is set to a specific value (line 2) then **Connect** produces a trajectory which arrives at  $\mu^b$  at time epoch  $k^b$  exactly.

## 6.2 Tree Shuffling

Shuffling is the process of probabilistically selecting a branch of the tree, detaching it from its parent and attaching it to another branch. The first step is to choose a node  $\eta^a$  at random. Then  $n_r$  other existing nodes are selected from the tree according

to a “fitness” function. Each of these nodes, denoted  $\eta^b \in \mathcal{T} \setminus \{\eta^a \cup \eta^0\}$ , are then disconnected from their current parents  $\eta^{\rho^b}$  and connected to  $\eta^a$  instead, as long as this switch lowers the resulting uncertainty cost of the subtree attached at  $\eta^b$  (see Fig. 7).

The *fitness density* over a given set  $\mathcal{T}$  is denoted  $q_{\mathcal{T}} : \mathcal{T} \rightarrow [0, 1]$  and defined by

$$q_{\mathcal{T}}(\eta^b) = \frac{\bar{q}_{\mathcal{T}}(\eta^b)}{\sum_{\eta \in \mathcal{T}} \bar{q}_{\mathcal{T}}(\eta)}, \text{ where } \bar{q}_{\mathcal{T}}(\eta^b) = e^{-\sqrt{\frac{j^b}{J_{\max}^*}}}, \quad (33)$$

where  $J_{\max}^*$  is a constant denoting the upper bound of an acceptable optimal cost that the algorithm is expected to yield. Sampling from the fitness function biases the selection of more capable nodes but without completely disregarding nodes with lower performance. This achieved by a distribution with a fat tail as shown on Fig. 6.

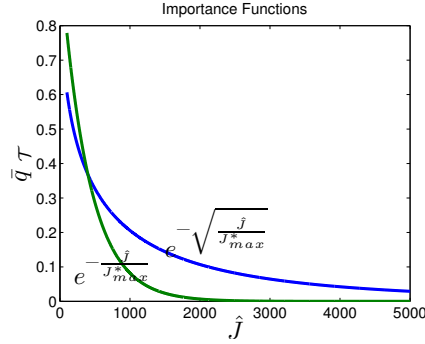


Figure 6: Two importance density functions  $\bar{q}_{\mathcal{T}}$  used to sample nodes during shuffling (with  $J_{\max}^* = 400$ ). The function with “fatter” tail (defined in (33)) is the proper choice in order to guarantee exploration of the state space.

Let the *subtree* rooted at  $\eta^b$  be denoted  $\mathcal{T}^b \subset \mathcal{T}$ . Define the combined trajectory connecting node  $\eta^a$  to node  $\eta^b$  and node  $\eta^b$  to node  $\eta^c$ , denoted  $\mu^{a \rightarrow b \rightarrow c}$ , by

$$\mu^{a \rightarrow b \rightarrow c} := \mu^{a \rightarrow b} \cup \mu^{b \rightarrow c}.$$

More precisely, a *shuffle*, i.e. parent switch  $\rho^b = a$ , occurs in two cases (see also Fig. 7). The obvious case is when the current optimal uncertainty cost  $\hat{J}^*$  can be improved by a trajectory  $\mu^{0 \rightarrow a \rightarrow b \rightarrow c}$  in the modified tree. The second case is heuristic: a switch occurs only if the cost can be lowered on average across all nodes in the subtree. These conditions are expressed as

$$\text{if } \left\{ \begin{array}{c} \min_{\eta^c \in \mathcal{T}^b} \hat{J}_{N|k^c}(\mu^{0 \rightarrow a \rightarrow b \rightarrow c}) < \hat{J}^* \\ \text{or} \\ \sum_{\eta^c \in \mathcal{T}^b} (\hat{J}_{N|k^c}(\mu^{0 \rightarrow a \rightarrow b \rightarrow c}) - \hat{J}^c) < 0 \end{array} \right\} \text{ then } \rho^b = a. \quad (34)$$

Note that  $\hat{J}^c$  in (34) should be understood as the present cost in the unmodified tree, i.e.  $\hat{J}^c := \hat{J}_{N|k^c}(\mu^{0 \rightarrow \rho^b \rightarrow b \rightarrow c})$ .

**Computational Savings.** The step (34) requires the computation of the uncertainty cost of all trajectories  $\mu^{0 \rightarrow a \rightarrow b \rightarrow c}$  obtained from the original  $\mu^{0 \rightarrow \rho^b \rightarrow b \rightarrow c}$  by replacing the segment  $\mu^{\rho^b \rightarrow b}$  with  $\mu^{a \rightarrow b}$ . Since all trajectories in the subtree at  $\eta^b$  are affected this could be an expensive operation. In addition, it seemingly requires that the densities (27) at node  $\eta^a$  be re-propagated along the complete and potentially long new trajectory segment  $\mu^{a \rightarrow b \rightarrow c}$ . In the SIS framework though it is not necessary to perform the whole propagation. In particular, only the incremental weight update  $U_{ij}$



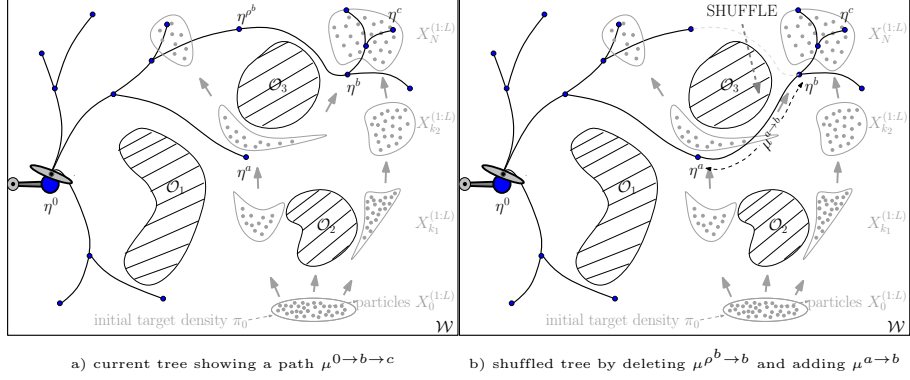


Figure 7: A tree *shuffling* iteration: a) a node  $\eta^a \in \mathcal{T}$  has been chosen at random; another node  $\eta^b \in \mathcal{T}$  is then selected with chance inversely proportional to its current uncertainty cost  $\hat{J}^b$ ; b)  $\eta^b$  is disconnected from its parent  $\eta^a$  and connected to  $\eta^b$  after checking that the uncertainty cost of the newly formed trajectory  $\mu^{0 \rightarrow a \rightarrow b \rightarrow c}$  either improves the global optimum  $J^*$  or on average improves the costs in the subtree  $\mathcal{T}^b$  (see (34)).

along the new segment  $\mu^{a \rightarrow b}$  must be computed (using (27)) and then the weights at all subtree nodes  $\eta^c \in \mathcal{T}^b$  are updated directly through the simple *weight rescaling* formula

$$\bar{W}_{ij}(\mu^{0 \rightarrow a \rightarrow b \rightarrow c}) = W_{ij}^c \frac{W_{ij}^a}{W_{ij}^b} U_{ij}, \quad (35)$$

where the weights  $W_{ij}^a$  are the existing weights at node  $\eta^a$ , resp.  $b$  and  $c$ . After computing the unnormalized weights (35) the cost  $\hat{J}_{N|k^c}(\mu^{0 \rightarrow a \rightarrow b \rightarrow c})$  used in the shuffling (34) is computed through (27b) and (28).

In summary, a shuffling step computes the incremental weights update along  $\mu^{a \rightarrow b}$  and simply rescales the existing weights at all affected child nodes  $\mathcal{T}^b$ . It is summarized according to

#### Shuffle

1. choose  $\eta^a$  from  $\mathcal{T}$  at random
2. for  $i = 1 : n_r$
3.    sample  $\eta^b \sim q_{\mathcal{T}}(\cdot)$
4.    execute (34)
5.    if  $\hat{J}^c < \hat{J}^*$  then  $\eta^* = \eta^c$ .

The number of nodes to be tried for a parent switch,  $n_r$ , during the shuffling step, can be constant but it is more reasonable to increase it as the tree becomes denser. Hence, the default choice used is  $n_r = \log(\dim(\mathcal{T}))$ .

### 6.3 Randomized Pruning

Shuffling §6.2 dynamically rebuilds the tree by removing and adding edges. A complementary operation can be considered, which dynamically adds and removes nodes based on their accumulated performance.

Denote the set of *leaf* nodes in tree by  $\mathcal{L}_{\mathcal{T}} \subset \mathcal{T}$ , i.e.

$$\mathcal{L}_{\mathcal{T}} := \{\eta^a \in \mathcal{T} \mid \nexists b \text{ s.t. } \rho^b = a\}.$$

Nodes are removed sequentially from the “bottom” of the tree, i.e. starting with leaf nodes. The procedure is summarized according to

#### Prune

1. for  $i = 1, \dots, n_p$
2.     $\mathcal{L}' = \mathcal{L}_{\mathcal{T}} \setminus \{\eta^*, \eta^0\}$

3.  $\eta \sim 1 - g_{\mathcal{L}'}(\cdot)$
4.  $\mathcal{T} = \mathcal{T} \setminus \{\eta\}$

Nodes to be pruned are selected inversely proportional to their fitness density (line 3). Empirically, as shown in §7, pruning turns out to be an effective strategy (again in the spirit of importance (re)sampling) for obtaining improved solutions quicker. Yet, the optimal choice for a number of nodes to be pruned  $n_p$  at every iteration is difficult to determine. In our tests we prune a small fraction of the total nodes  $n$ , i.e.  $n_p = n/5$ .

## 6.4 Rate of Convergence

The success of the proposed algorithm depends on two key issues—whether it converges to an optimum and the rate at which it converges. A basic requirement is to obtain asymptotic convergence, i.e. to reach the optimal as the number of algorithm iterations tends to infinity.

The random expansion algorithm (RND) (§5.1) samples points and connects them uniformly at random. This is equivalent to generating random trajectories covering the search space. The motion planning approach ensures that the tree trajectories satisfy the dynamics and constraints. In principle, the algorithm will find a trajectory close to the optimal as the number of iterations increases. Yet, this might be an infinitely slow process in practice, a fact also confirmed by simulations in the next section §7.

Utility-based importance sampling (§6.1) can provide a good solution quicker by biasing trajectories to pass through more promising parts of the state space. Further work is necessary to establish non-asymptotic convergence rates by assuming certain problem structure and regularity conditions. Currently, the advantage of this proposed technique is observed empirically for specific problems.

The idea of shuffling (§6.2) is to reach close to an optimum not by generating completely new trajectories (as RND does) but instead to perform local modifications to the existing tree structure. Assume that the optimal trajectory which the algorithm seeks to compute consists of  $m+1$  nodes  $\eta^0, \bar{\eta}^1, \dots, \bar{\eta}^m$ . As the number of sampled nodes in the tree  $\mathcal{T}$  increases there will be  $m$  nodes  $\hat{\eta}^i \in \mathcal{T}$  approaching asymptotically close to each of the unknown optimal nodes  $\bar{\eta}^i$ ,  $i = 1, \dots, m$ . The problem is that there is a very small probability that all  $\hat{\eta}^i$  will in fact be connected by a physical path contained in the tree, i.e. a path which itself is approaching the optimal path. The purpose of shuffling is then to remove and add edges in an attempt to discover this path.

Shuffling becomes less effective as the number of nodes increases since the number of pairs of old and new subtree parents to be tested for shuffling grows quadratically. The purpose of pruning (§6.3) is then to remove less promising nodes in order to reduce the search space. In essence, the size of the solutions space depends on the number of nodes and on the edges connecting these nodes. Shuffling allows control over the set of edges while sampling and pruning dynamically control the set of nodes. Empirical studies of the convergence rates are studied next.

## 7 Numerical Tests

Numerical studies based on the simple vehicle are presented first followed by a more complex helicopter search example.

### 7.1 Simple Vehicle

The methods are tested through multiple simulation runs in the simulated scenario defined in §2 with tree node connection defined in §5.2. Four algorithms are developed and analyzed in a sequence in order to compare the proposed baseline algorithm and variance reduction techniques. The algorithms are abbreviated according to:

- RND: baseline random expansion algorithm (§5.1)

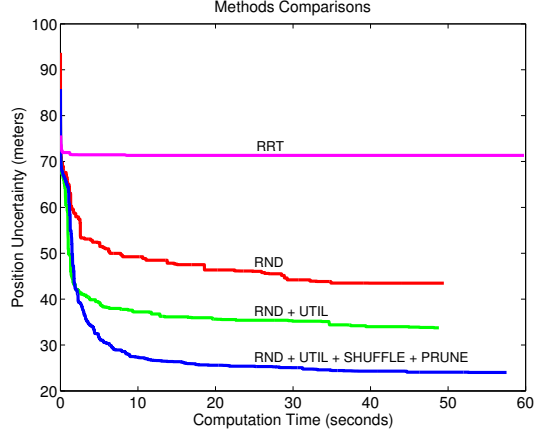


Figure 8: Comparison of several tree search algorithms. A standard RRT algorithm is not suitable for optimal planning since it quickly converges to and remains at a low quality solution. The algorithm RND converges asymptotically but the rate decreases with computation time. Utility-based sampling (UTIL) speeds up convergence but not drastically. The final complete algorithm including shuffling and pruning provides the best performance providing an acceptable error below 25 meters (i.e. the y-axis plots the square root of the total variance  $\sqrt{\hat{J}^*}$  which can be interpreted as a combined error in the two position coordinates)

- RND+UTIL: RND augmented with utility-based sampling (§6.1)
- RND+UTIL+SHUFFLE: with the addition of a shuffling step (§6.2)
- RND+UTIL+SHUFFLE+PRUNE: the final algorithm including pruning (§6.3)

Fig. 8 shows the resulting averaged results of the performance of each algorithm, including a comparison with a standard RRT expansion. For completeness, more detailed plots are also given in Fig. 10. The simulations shows that a random search tree (RND) is more suitable than a standard motion planning tree for obtaining convergence to an optimal trajectory. Yet the convergence is very slow. This is remedied by the combination of the proposed variance reduction techniques. The complete algorithm RND+UTIL+SHUFFLE+PRUNE computes a solution with an acceptable performance within the allotted computation time of 60 seconds. Several snapshots of the dynamic tree and the resulting optimal trajectory as the algorithm progresses are shown on Fig. 9. Note that all reported results are based on a global open-loop optimization with simulated future measurements rather than in a receding horizon fashion.

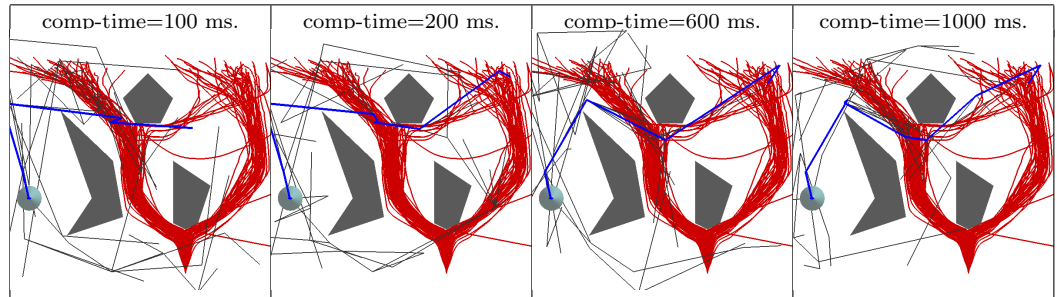


Figure 9: Several frames showing the current tree and optimal trajectory computed by algorithm RND+UTIL+SHUFFLE+PRUNE.

Finally, it should be noted that the complete algorithm requires several parameters which must be selected in advance. They are listed in the following table:

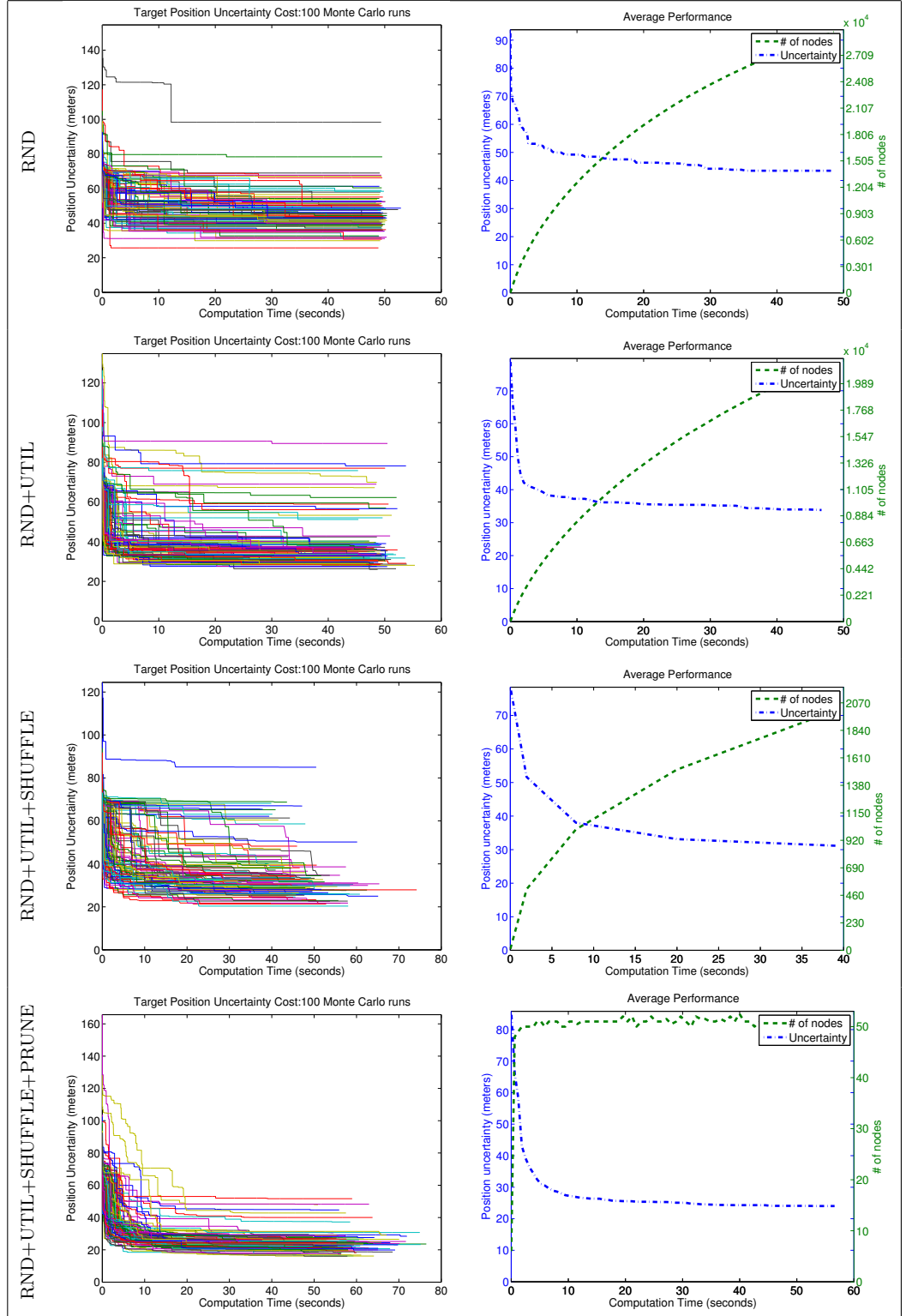


Figure 10: Monte Carlo analysis of the the proposed techniques applied to the scenario in §2. The performance metrics are the resulting optimal uncertainty cost  $\sqrt{\hat{J}^*}$  (which can be regarded roughly as the standard deviation of the distance to the true value) and the corresponding number of tree nodes. These two metrics are expressed in terms of the required computational time (shown on  $x$ -axis). The left plots show all 100 Monte Carlo runs while the right shows the averaged results.

Param.	Description	Default
$n_i$	# of nodes to be added at iteration $i$ (i.e. $\dim(\mathcal{T}) = \sum n_i$ )	10
$P_U$	utility-sampling probability	.5
$n_r$	# of nodes checked for shuffling at iteration $i$	$\log(\dim(\mathcal{T}))$
$n_p$	# of nodes to be pruned at iteration $i$	$\dim(\mathcal{T})/5$

The optimal values of these parameters are difficult to determine and might vary as the optimization runs. This is a problem that requires further study. The recommended default values are chosen to provide a balance between the baseline random tree that explores the state space and the variance reduction steps to focus and speed-up the search.

## 7.2 A Helicopter Search Scenario

Consider a small autonomous helicopter depicted in Fig. 11 operating in a 3-D terrain. The vehicle is modeled as a single underactuated rigid body with position  $r \in \mathbb{R}^3$  and orientation  $R \in \text{SO}(3)$  where  $\text{SO}(3)$  denotes the space of right-handed coordinate frames described by three orthogonal vectors (i.e. by a 3x3 orthogonal matrix with positive determinant). Its *body-fixed* angular and linear velocities are denoted by  $\omega \in \mathbb{R}^3$  and  $v \in \mathbb{R}^3$ , respectively. The vehicle has mass  $m$  and principal moments of rotational inertia  $J_1, J_2, J_3$  forming the inertia tensor  $\mathbb{J} = \text{diag}(J_1, J_2, J_3)$ .

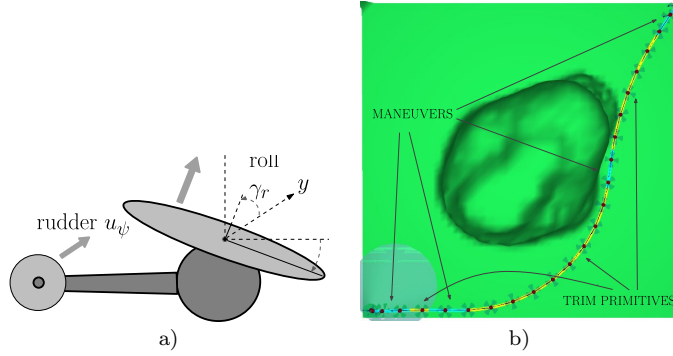


Figure 11: Simplified helicopter model used in our tests (a). An example of a computed sequence of four maneuvers and three trim primitives is shown (b), connecting two zero-velocity states in the corners of the workspace and avoiding an obstacle in the center. The trim velocities satisfy invariance conditions defined in §B.1 while the maneuvers are computed as outlined in §B.2.

The vehicle is controlled through a *collective*  $u_c$  (lift produced by the main rotor) and a *yaw*  $u_\psi$  (force produced by the rear rotor), while the direction of the lift is controlled by tilting the main blades forward or backward through a *pitch*  $\gamma_p$  and sideways through a *roll*  $\gamma_r$ . The four control inputs then consist of the two forces  $u = (u_c, u_\psi)$  and the two shape variables  $\gamma = (\gamma_p, \gamma_r)$ . The state space of the vehicle is  $\mathcal{M} = \text{SO}(3) \times \mathbb{R}^3 \times \mathbb{R}^6 \times \mathbb{R}^2$  with  $\mu = ((R, p), (\omega, v), \gamma)$ .

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} R \hat{\omega} \\ R v \end{bmatrix}, \quad (36)$$

$$\begin{bmatrix} \mathbb{J} \dot{\omega} \\ m \dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{J} \omega \times \omega \\ m v \times \omega + R^T(0, 0, -9.81m) \end{bmatrix} + F(\gamma)u, \quad (37)$$

where the map  $\hat{\cdot}: \mathbb{R}^3 \rightarrow \mathfrak{so}(3)$  is defined by

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix},$$

while the control matrix is defined as

$$F(\gamma) = \begin{bmatrix} d_t \sin \gamma_r & 0 \\ d_t \sin \gamma_p \cos \gamma_r & 0 \\ 0 & d_r \\ \sin \gamma_p \cos \gamma_r & 0 \\ -\sin \gamma_r & -1 \\ \cos \gamma_p \cos \gamma_r & 0 \end{bmatrix}.$$

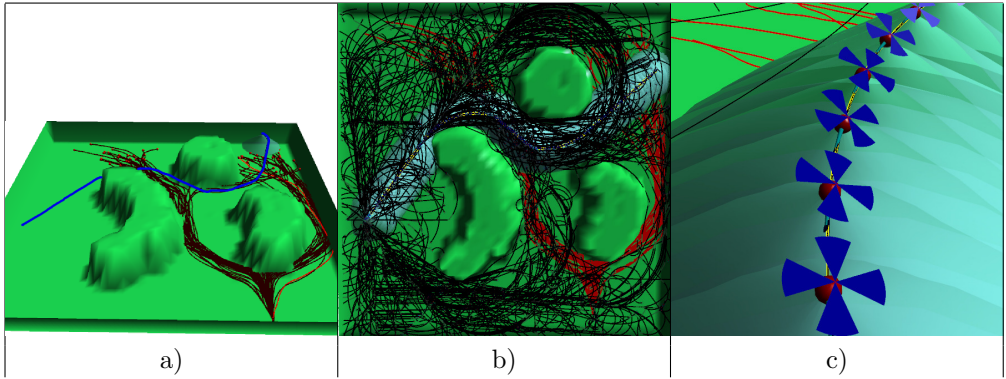


Figure 12: The algorithm applied to the helicopter example described in §7.2 showing a) the optimal helicopter trajectory; b) the constructed roadmap and sensor footprint along a path; c) close-up view along an edge of the roadmap.

The local motion planning method corresponding to **Connect** is based on sequencing of precomputed motion primitives which satisfy the dynamics (36)–(37). This is accomplished using a maneuver automaton described in §B. In essence, the set of primitives abstracts away the complex dynamics and reduces the edge creation problem to an optimization in the discrete set of primitives and the space of translations and planar rotations –  $SE(2) \times \mathbb{R}^1$ . A trajectory consisting of a given sequence of minimum number of primitives can then be computed instantly in closed form through inverse kinematics. Fig. 11 shows an example of such a sequence of primitives connected in order to exactly solve the boundary value problem. The terrain is represented using a digital elevation map loaded from a file. Collision checking and avoidance is performed using the Proximity Query Package (PQP) (Gottschalk et al., 1996) that compute closest distance between arbitrary polyhedra and is used to implement the function **prox** defined in (5).

The algorithm is tested in scenario similar to §2 extended to 3-D. The helicopter is not permitted to fly above obstacles. Fig. 12 shows the resulting helicopter trajectory (see also video in Extension 1 ( §A)), a view of the constructed roadmap, and a close-up along the optimal path within the roadmap.

## 8 Conclusion

This work deals with optimal estimation for systems with nonlinear dynamics subject to nonconvex constraints. The approach is based on a random enumeration of trajectories generated from a tree which compactly approximates the reachable space and efficiently propagates probability distributions through recursion. The randomly sampled tree nodes approach any reachable state with exponentially (in the number iterations) high probability and therefore encode a versatile roadmap of solution trajectories. Yet, without assuming any special structure known a-priori, random search alone does not result in an efficient algorithm due to the high-dimensionality of the problem. This

issue is alleviated through variance reduction techniques similar to importance sampling for stochastic optimization and to cross-over in evolutionary algorithms. While these methods show a marked improvement in solution quality and run-time efficiency, no formal non-asymptotic convergence rates have been established. A possible future direction is to address this issue by assuming certain regularity conditions on the models involved. A related direction is to combine the proposed approach with the *cross-entropy* (CE) optimization method (Rubinstein and Kroese, 2004; Celeste et al., 2007) which is designed to explicitly identify structure in the solution space by maintaining and optimally adapting an importance sampling distribution. Guiding the random tree expansion through a CE-type method would provide a consistent exploration–exploitation approach (for initial developments see (Kobilarov, 2011)) that optimally accounts for the sampled data during optimization. Finally, even though formally fast convergence rates are absent in our general setting, this work provided a simple particle-based algorithm applicable to general types of dynamics and uncertainty models which is easy to implement and performs well in practice.

## Acknowledgements

The authors thank the reviewers for their comprehensive review and advise for improving the paper quality.

## Appendix

### A Index to Multimedia Extensions

The multimedia extensions to this article are at: <http://www.ijrr.org>.

Extension	Type	Description
1	Video	Helicopter Search Scenario

### B Helicopter Primitives

The local motion planning method is based on computing a sequence of *motion primitives* that exactly satisfy the boundary conditions, i.e., exactly reaches a sampled node. The symmetry in the system dynamics allows us to employ a *maneuver automaton* to produce sequences of continuously parametrizable motions (trim primitives) connected with maneuvers. This general framework developed in (Frazzoli et al., 2005) is suitable to systems such as UAVs or ground robots if one ignores pose-dependent external forces, such as varying wind or changing ground friction as function of position.

Let the vehicle rotation be described by its roll  $\phi$ , pitch  $\theta$ , and yaw  $\psi$ . Denote the linear velocity by  $v = (v_x, v_y, v_z) \in \mathbb{R}^3$ , and the angular velocity by  $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ . Denote the whole configuration by  $g \in SE(3)$ , the whole velocity by  $\xi_b \in \mathfrak{se}(3)$ , where  $SE(3)$  and  $\mathfrak{se}(3)$  denote the Euclidean group (translations and rotations) and its set of body-fixed velocities, respectively. The elements are defined by

$$g = \begin{bmatrix} R & r \\ \mathbf{0} & 1 \end{bmatrix}, \quad \xi_b = \begin{bmatrix} \hat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix}. \quad (38)$$

By defining the map  $\mathbb{I} = \text{diag}(J_1, J_2, J_3, m, m, m)$  the dynamics can be expressed in more general form as

$$\mathbb{I}\dot{\xi}_b = \text{ad}_{\xi_b}^* \mathbb{I}\xi_b + f_u + \text{Ad}_g^* f_{ext},$$

where  $f_u$  is the control force, corresponding to  $F(\gamma)u$  in (37), while  $f_{ext} = (0, 0, 0, 0, 0, -9.81m) \in \mathfrak{se}(3)^*$  is the gravity force. Since gravity is the only configuration-dependent term in the

dynamics and is invariant to translations and rotations the  $z$ -axis, then the dynamics symmetry group can be set as  $G = SE(2) \times \mathbb{R}$ .

The motion planning problem is solved in closed form through inverse kinematics of a minimal number of primitives. A total of five is employed when moving between non-equilibrium states and an extra maneuver is added to connect from/to an equilibrium (zero velocity) state. Our particular design of the trims and maneuvers used is described next. An example sequence is shown of Fig. 11.

## B.1 Trim Primitives

Denote the transformation corresponding to roll and pitch only by  $g(\phi, \theta) \in SE(3) \setminus G$ . The  $G$ -symmetry corresponds to velocity  $\xi \in \mathfrak{se}(2) \times \mathbb{R}$ , that corresponds to velocity vector  $(0, 0, \omega_z, v_x, v_y, v_z)$ , for which  $\xi_b = \text{Ad}_{g(\phi, \theta)}^{-1} \xi$  is a relative equilibrium for the whole system on  $SE(3)$ , i.e.,  $\dot{\xi}_b = 0$  and  $g(t) = g(0) \exp(t\xi_b)$ . This velocity is obtained by satisfying this invariance condition, or equivalently

$$\text{ad}_{\xi_b}^* \mathbb{I} \xi_b + f_u + \text{Ad}_{g(\phi, \theta)}^* f_{ext} = 0 \quad (39)$$

since  $f_{ext}$  is  $G$ -invariant.

The conditions (39) can be simplified if one assumes that the moments of rotational inertia around the  $y$  and  $z$  axis are identical. In this case the invariance conditions simplify to:

$$\begin{aligned} \theta &= 0, & u_y &= 0, & \gamma_p &= 0, & \gamma_r &= 0, \\ \phi &= \arctan(-w_z v_x / 9.81), \\ u_c &= m(\cos \phi \, 9.81 - w_z v_x \sin \phi). \end{aligned} \quad (40)$$

In order to design trim primitives, one can pick desired velocities  $(\omega_z, v_x, v_y, v_z)$  and use (40) to compute the required constant roll, pitch, and control inputs for motion along that trim. Since control inputs and shape variables have bounds, the equations (40) can also be inverted to compute the maximum sustainable trim velocities.

## B.2 Maneuvers

Maneuvers are computed to connect two trim motions. The parameters of a trim primitive are its roll, pitch, velocities, and shape variables. Let the map  $\pi : X \rightarrow X \setminus G$  subtract out the invariant coordinates from a given state. Then given two trims, the first one ending with state  $x_1$  and the second one starting with state  $x_2$ , we compute a maneuver trajectory  $x$  using the following optimization procedure:

$$\begin{aligned} \textbf{Compute:} & \quad T; \, x : [0, T] \rightarrow X; \, u : [0, T] \rightarrow U \\ \textbf{minimizing:} & \quad J(x, u, T) = \int_0^T 1 + \lambda \|u(t)\|^2 \, dt \\ \textbf{subject to:} & \quad \pi(x(0)) = x_1, \pi(x(T)) = x_2, \\ & \quad \text{dynamics eq. (36),(37) for all } t \in [0, T]. \end{aligned}$$

The parameter  $\lambda$  controls the importance of minimizing the control effort. An alternative strategy is to fix  $T$  in the above formulation and to search for the minimal  $T$  yielding a feasible control-effort optimal problem. We perform this using binary search for  $T$  over the real line by solving a fixed- $T$  optimal control problem in each iteration.

The continuous optimal control formulation is computationally solved through the discrete mechanics methodology (Marsden and West, 2001) which is particularly suitable for systems with nonlinear state spaces and symmetries. A geometric structure preserving optimizer developed in (Kobilarov, 2008, Sec. §2.7) is used to perform the computations. The computations are performed offline with the resulting optimal maneuvers assembled in a library offering instant lookup during run-time.



## References

- Bethke, B., Bertuccelli, L. F., and How, J. P. (2008). Experimental demonstration of adaptive MDP-based planning with model uncertainty. In *AIAA Guidance, Navigation and Control Conference*.
- Bryson, M. and Sukkarieh, S. (2009). Architectures for cooperative airborne simultaneous localisation and mapping. *Journal of Intelligent and Robotic Systems, Special Issue on Airborne SLAM*, 55:267–297.
- Burns, B. and Brock, O. (2005). Toward optimal configuration space sampling. In *Robotics: Science and Systems*.
- Celeste, F., Dambreville, F., and Cadre, J.-P. L. (2007). Optimal path planning using cross-entropy method. In *2006 9th International Conference on Information Fusion*, pages 1 – 8.
- Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.
- Chung, T. H. and Burdick, J. W. (2007). A decision-making framework for control strategies in probabilistic search. In *Intl. Conference on Robotics and Automation*, pages 4386–4393. ICRA.
- Clarke, F. H., Ledyaev, Y. S., Stern, R. J., and Wolenski, P. R. (1998). *Nonsmooth Analysis and Control Theory*. Springer.
- Cole, D. T., Goktogan, A. H., and Sukkarieh, S. (2008). The demonstration of a cooperative control architecture for UAV teams. In *Experimental Robotics*, volume 39, pages 501–510. Springer Berlin / Heidelberg.
- de Geeter, J., de Schutter, J., Bruyninckx, H., van Brussel, H., and Decreton, M. (1998). Tolerance-weighted L-optimal experiment design for active sensing. In *Proc. Conf. IEEE/RSJ Int Intelligent Robots and Systems*, volume 3, pages 1670–1675.
- Del Moral, P. (2004). *Feynman-Kac Formulae. Genealogical and interacting particle systems with applications*. Springer-Verlag.
- Doucet, A., de Freitas, N., and Gordon, N., editors (2001). *Sequential Monte Carlo methods in practice*.
- Erinc, G. and Carpin, S. (2007). A genetic algorithm for nonholonomic motion planning. In *Proc. IEEE Int Robotics and Automation Conf*, pages 1843–1849.
- Frazzoli, E., Dahleh, M. A., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129.
- Frazzoli, E., Dahleh, M. A., and Feron, E. (2005). Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091.
- Geyer, C. (2008). Active target search from UAVs in urban environments. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, pages 2366–2371.
- Gottschalk, S., Lin, M. C., and Manocha, D. (1996). OBBTree: A hierarchical structure for rapid interference detection. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 30:171–180.

- Grocholsky, B., Makarenko, A., and Durrant-Whyte, H. (2003). Information-theoretic coordinated control of multiple sensor platforms. In *Proc. IEEE International Conference on Robotics and Automation ICRA '03*, volume 1, pages 1521–1526.
- He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008)*, pages 1814–1820, Los Angeles, CA.
- Hocaoglu, C. and Sanderson, A. C. (2001). Planning multiple paths with evolutionary speciation. *IEEE Transactions on Evolutionary Computing*, 5(3):169–191.
- Hoffmann, G. M. and Tomlin, C. J. (2009). Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*.
- Hollinger, G., Singh, S., and Kehagias, A. (2009). Efficient, guaranteed search with multi-agent teams. In *Robotics: Science and Systems*.
- Hsu, D., Kindel, R., Latombe, J., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robotics Research*, 21(3):233–255.
- Kobilarov, M. (2008). *Discrete Geometric Motion Control of Autonomous Vehicles*. PhD thesis, University of Southern California.
- Kobilarov, M. (2011). Cross-entropy randomized motion planning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA.
- Krause, A. and Guestrin, C. (2007). Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *International Conference on Machine Learning (ICML)*, Corvallis, Oregon.
- Ladd, A. and Kavraki, L. (2004). Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics and Automation*, 20(2):229–242.
- Langdon, W. B. and Poli, R. (2001). *Foundations of Genetic Programming*. Springer.
- Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Press.
- LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K.
- Lavis, B., Furukawa, T., and Whyte, H. F. D. (2008). Dynamic space reconfiguration for Bayesian search and tracking with moving targets. *Auton Robot*, 24:387399.
- Marsden, J. and West, M. (2001). Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514.
- Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. (2009). A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Auton Robot*, 27:93103.
- Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput.*, 4:1–32.
- Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., and Schutter, J. D. (2003a). A comparison of decision making criteria and optimization methods for active robotic sensing. In *Lecture Notes in Computer Science*, volume LNCS 2542, pages 316–324.
- Mihaylova, L., Schutter, J. D., and Bruyninckx, H. (2003b). A multisine approach for trajectory optimization based on information gain. *Robotics and Autonomous Systems*, 43(4):231–243.

- Paris, S. and Le Cadre, J.-P. (2002). Planning for terrain-aided navigation. In *Proc. Fifth Int Information Fusion Conf*, volume 2, pages 1007–1014.
- Powell, W. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics.
- Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer.
- Rubenstein, R. Y. and Kroese, D. P. (2008). *Simulation and the Monte Carlo Method*. Wiley.
- Rubinstein, R. Y. and Kroese, D. P. (2004). *The cross-entropy method: a unified approach to combinatorial optimization*. Springer.
- Ryan, A. (2008). Information-theoretic tracking control based on particle filter estimate. In *AIAA Guidance, Navigation, and Control Conf*.
- Sim, R. and Roy, N. (2005). Global A-optimal robot exploration in slam. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2005*, pages 661–666.
- Singh, S. S., Kantas, N., Vo, B.-N., Doucet, A., and Evans, R. J. (2007). Simulation-based optimal sensor scheduling with application to observer trajectory planning. *Automatica*, 43:817–830.
- Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems*.
- Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotic*. MIT Press.
- Tisdale, J., Kim, Z., and Hedrick, J. (2009). Autonomous UAV path planning and estimation. *IEEE Robotics & Automation Magazine*, 16(2):35–42.
- Tremois, O. and Le Cadre, J.-P. (1999). Optimal observer trajectory in bearings-only tracking for maneuvering sources. *IEE Proceedings -Radar, Sonar and Navigation*, 146(1):31–39.
- Vaidyanathan, R., Hocaoglu, C., Prince, T. S., and Quinn, R. D. (2001). Evolutionary path planning for autonomous air vehicles using multi-resolution path representation. In *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, volume 1, pages 69–76.
- Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation*, 1(1):18–28.