The International Journal of
Robotics Research
31(1) 24–41

DOI: 10.1177/0278364911423558
ijr.sagepub.com

Global estimation in constrained environments

Marin Kobilarov[1], Jerrold E Marsden[1] and Gaurav S Sukhatme[2]

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 that minimizes the estimation error over a given time horizon taking into account uncertainties in the observable dynamics and 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 and a simulated helicopter searching for a moving target in a 3-D terrain.

Keywords

Aerial robotics, motion planning, estimation, search and rescue robots

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. This sort of capability is required, for instance, for time-critical *surveillance* or *search-and-rescue* missions.

The problem is formally defined through a hidden Markov model (HMM) of a stochastic process $\{(X_k, Y_k)\}_{0\le k\le N}$ where X_k denotes the hidden target state and Y_k denotes the observation at the kth 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 by

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

$$Y_k = g(X_k, V_k; \mu_k), \tag{1b}$$

where Ω_k and V_k are independent and identically distributed (i.i.d.) noise terms and $\mu_k \in \mathcal{M}$ denotes the vehicle state[1] at time k. The manifold $\mathcal{M}$ need not be a vector space. A trajectory of states between two epochs i and j, where $0 \le i < j \le N$, is denoted by $x_{i:j} := \{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$.

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

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

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

$$h_c(\mu_k) \ge 0, \text{ for all } 0 \le k \le N. \tag{3}$$

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 the one hand, complicated non-linear dynamics and obstacles induce multiple homotopy classes of vehicle trajectories that preclude convexity. On the other, function (2) can have singularities (and, hence, might not be smooth everywhere) when the dynamics is underactuated or non-holonomic (Choset et al. 2005; LaValle 2006). In addition, the distance-to-obstacle

[1]California Institute of Technology, Pasadena, CA, USA
[2]University of Southern California, Los Angeles, CA, USA

Corresponding author:
Marin Kobilarov, 2543 Wellesley Avenue, Los Angeles, California 90064, US.
Email: marin@cds.caltech.edu

function [encoded in (3)] might not be differentiable, for example, 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 numerically computes 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 $y_{1:k}$.

1.1. 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 $\varphi : \mathcal{X} \to \mathbb{R}^{n'_x}$, where $n'_x \leq n_x$, selects and weighs a combination of these coordinates. For instance, φ 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 by

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

subject to the dynamics (2) and constraints (3). The expectation in (4) is taken over all future realizations of the states $X_{0:N}$ and the measurements $Y_{1:N}$ while π_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 the 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 weighting matrix M corresponds to a well-established tolerance-weighted error or L-optimal design (de Geeter et al. 1998).

1.2. Simple example

These definitions can be illustrated with a simple example of a target modeled as a unit-mass particle moving in a plane. 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 ω and ν are realizations of, for example, 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 meters then the square root of the right-hand side of (4) is also in meters, which is convenient for establishing meaningful tolerances.

1.3. 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 for the target-tracking community. It is also highly relevant to the problem of *active sensing* studied in robotics (Grocholsky et al. 2003; Mihaylova et al. 2003a; Thrun et al. 2005) where the vehicle estimates its own state (Paris and Le Cadre 2002; He et al. 2008) and in some cases refines its knowledge about the environment (Sim and Roy 2005; Stachniss et al. 2005).

One approach is to solve the problem approximately by discretizing the vehicle and target state spaces. Such techniques, for example, 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, a policy search in an information space (LaValle 2006) or a Markov decision process (MDP)-based search [e.g. Bethke et al. (2008)] would typically be based on such representations. We emphasize that our main interest is in high-dimensional problems dominated by fast non-linear and underactuated dynamics and sensing. In this context techniques exploiting convexity [respectively submodularity (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 or Gaussian structure; they are limited to myopic one-step optimal decision making; or 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 it exploits the 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, that is, using Gaussian process cost-function approximation to speed up 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; Bryson and Sukkarieh 2009; Hoffmann and Tomlin 2010), or unconstrained scenarios (Geyer 2008; Ryan 2008).

1.4. Overview of contributions and approach

The distinctive feature of this paper is the treatment of the constraints (2) and (3) in the optimization (4). In particular, gradient-based optimization as in Paris and Le Cadre (2002), Mihaylova et al. (2003b), and 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 paper 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, for example, 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 paper 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 (Rubinstein and Kroese 2008) it is essential to exploit the 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 the 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) and (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 on 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 Table 1 and will be developed in detail in the paper.

Table 1. Variance-reduction techniques.

Technique	Exploration	Exploitation	Computational efficiency
Tree expansion	√		
Biased sampling	√	√	
Shuffling		√	√
Pruning			√

1.5. 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* using the '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, such as penalty functions or infeasible path rejection, employed in works such as Xiao et al. (1997), Vaidyanathan et al. (2001), Hocaoglu and Sanderson (2001), and 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 paper 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 so that there is convergence to an optimal trajectory. While shuffling and pruning might seem akin to standard genetic operation,

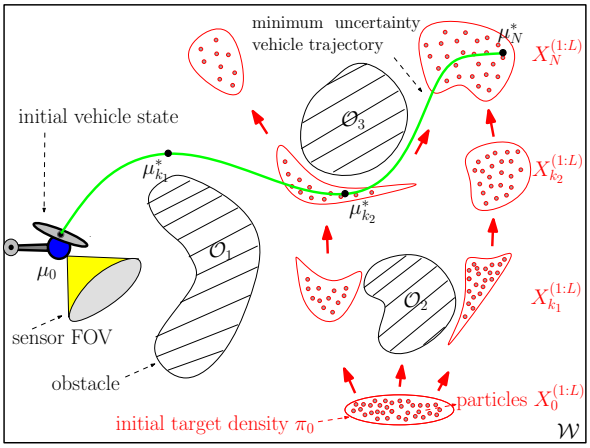


Fig. 1. A scenario with a vehicle (depicted as a small helicopter) at state $\mu_0 \in \mathcal{M}$ and a target with initial distribution π_0 diffusing north. Both target and vehicle avoid obstacles denoted $\mathcal{O}_i$. The set of possible target motions is approximated by L sampled trajectories $X_{0:N}^{(\ell)}$ for $\ell = 1, \ldots, L$. The figure shows the sampled states (particles) at the beginning $k = 0$, at two intermediate times $0 \le k_1 \le k_2 \le N$, and at the horizon $k = N$. We wish 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.

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

Consider the scenario depicted in Figure 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 $\mathcal{O}_1, \ldots, \mathcal{O}_{n_o} \subset \mathcal{W}$, which the vehicle must avoid.

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, for example, 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 to avoid obstacles, generally expressed as

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

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 Section 7.2.

2.1. A simple vehicle

Consider a point-mass vehicle moving in a 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

$$r_{k+1} = r_k + \tau v_k, \tag{6}$$

which is encoded by the function h_d defined in (2). The constant τ is the *time step*, that is, 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 (Figure 1) a bound of $v_{\max} = 8$ m s^{-1} 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 s.

2.2. Target dynamics

The target is modeled as a point mass on the ground with position $r = (r_x, r_y) \in \mathbb{R}^2$ and 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 a white-noise acceleration component Ω with standard deviations σ_x and σ_y. The model is

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

where τ is the time step; $K_p > 0$ and $K_d > 0$ are the 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; and $d_k = \|g_k\|$ 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 parameters. For instance, in the numerical scenarios studied in this paper we use

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

where β_k is the 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 $r_y = 200$.

2.3. Sensor model

The vehicle is equipped with sensors, which provide the relative range and bearing to the 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

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

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

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

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

$$g((r,v);(r',v'),V) = \begin{cases} \left.\begin{array}{ll} g^*((r,v);(r',v')) + \frac{\|r-r'\|}{d_s} V, & r \in \mathcal{V}(r') \\ \emptyset, & r \notin \mathcal{V}(r') \end{array}\right\} & \text{if } u_f \geq P_f \\ g^*((r^f,v);(r',v')),\ r^f \sim \mathcal{U}(\mathcal{V}(r')) & \text{if } u_f < P_f, \end{cases} \tag{9}$$

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

2.4. 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

$$X_0 \sim \pi_0, \tag{10a}$$
$$X_k \sim p(\cdot|X_{k-1}), \qquad k > 0 \tag{10b}$$
$$Y_k \sim q(\cdot|X_k;\mu_k), \qquad k > 0 \tag{10c}$$

where π_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

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

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

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

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 $\Phi_{N|k} : \mathcal{Y}^k \times \mathcal{M}^k \to \mathbb{R}^{n'_x}$ and defined by

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

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 $J_{N|k} : \mathcal{M}^k \to \mathbb{R}$ and defined as

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

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

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

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

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

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 non-linear or non-Gaussian models. Therefore, following Singh et al. (2007), we employ a particle-based approximation using L delta distributions, placed at state *samples* $X_k^{(j)} \in \mathcal{X}$ with positive weight functions $w_k^{(j)} : \mathcal{Y}^k \times \mathcal{M}^k \to \mathbb{R}_+$, that is

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

where δ_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, that is, $\sum_{j=1}^{L} w_k^{(j)} = 1$. This is equivalent to a sequential importance sampling (SIS) scheme with importance distribution $\pi_0(x_0)\,\Pi_{i=1}^{k} p(x_i|x_{i-1})$. Note that while more sophisticated sampling methods have been developed, for example, that additionally account for measurements to reduce variance (Doucet et al. 2001; Robert and Casella 2004), this paper follows the basic choice for simplicity. Figure 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, such as $N < 30$. 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), that is by drawing $\left(X_{0:N}^{(\ell)}, Y_{1:k}^{(\ell)}\right)$ 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 $i = 1,\ldots,k$. 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, that is

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

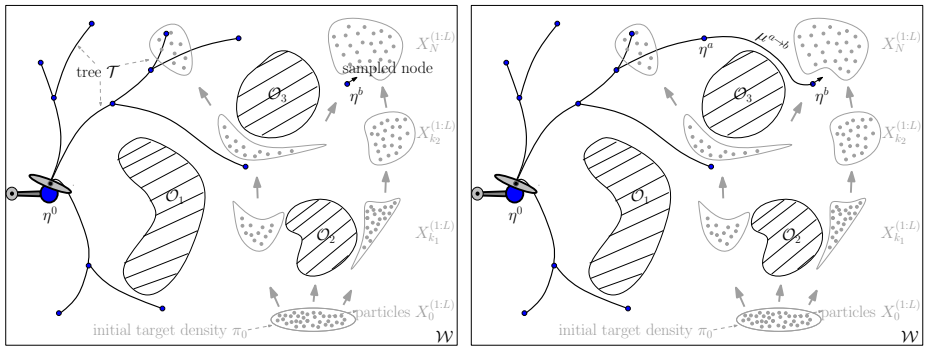
with $\hat{J} := \hat{J}_{N|N}$ denoting the approximate cost of a whole trajectory $\mu_{1:N}$. The global optimization algorithms developed in this paper will be based on the approximate estimate (21) and cost (22), that is, they will solve

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

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 $\mathcal{M}$, for example, using a grid and generating candidate paths by transitioning between adjacent cells. Such an approach is generally limited to a few dimensions, such as $\dim(\mathcal{M}) \leq 3$, and to



(a) A tree $\mathcal{T}$ and a newly sampled node η^b (b) Connecting η^b to the tree

Fig. 2. A tree-expansion step implemented by **Expand** (Section 5.1): (a) a node η^b is sampled; (b) it is then connected to a randomly chosen node $\eta^a \in \mathcal{T}$.

systems with very simple dynamics, for example 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 a tree-based search but unlike a 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 road map* (PRM) (Choset et al. 2005). Unlike these motion-planning algorithms, the trees employed in this paper 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 Section 6 introduces the variance-reduction techniques that complete the overall approach.

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 $0 \le k \le N$; the vehicle state μ; the particle weights matrix W, which gives a convenient way to compute the density $\hat{\pi}$; the target state estimate uncertainty cost $\hat{J} \ge 0$; and the tree parent index ρ. Nodes and their subelements are indexed by superscripts, that is, η^a has state μ^a and its parent node is η^{ρ^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 $W^0_{\ell j} = 1/L$ for all $\ell, j = 1, \ldots, L$. A *trajectory* between two nodes, η^a and η^b, is denoted $\mu^{a\to b}$ and a state at time k along this trajectory is denoted $\mu_k^{a\to b}$ where $k^a \le k \le k^b$.

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 $\mu^{0\to a}$. An overview of the elements comprising each node is given in Table 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 η^a and η^b. For instance, if the states μ^a and μ^b were close enough and no obstacles between them were present then a trajectory $\mu^{a\to b}$ would be produced, otherwise the connection fails. Such a controller is abstractly represented by the function **Connect**, that is

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

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

$$\eta^b = \textbf{Sample}(). \tag{25}$$

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

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

One of these nodes denoted $\eta^a \in \mathcal{T}^{\to b}$ is selected uniformly at random to become the parent of η^b linked with trajectory $\mu^{a\to b}$, that is $\rho^b = a$. Figure 2 illustrates the construction.

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

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

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

Variable	Type	Element description
k^b	integer ≥ 0	time epoch index
μ^b	$\dim(\mathcal{M}) \times 1$ vector	vehicle state at node
W^b	$L \times L$ matrix	weights $W^b_{\ell j} := w^{(j)}_{k^b}(Y^{(\ell)}_{1:k^b}; \mu^{0\to b})$, for $\ell, j = 1, \ldots, L$
$\hat{J}^b$	scalar > 0	uncertainty cost $\hat{J}^b := \hat{J}_{N\mid k^b}(\mu^{0\to b})$
ρ^b	integer ≥ 0	parent-node index

for all $\ell = 1, \ldots, L$. A row in the matrix W^b, that is, W^b_ℓ for any $1 \leq \ell \leq L$, corresponds to the resulting weights for each measurement sequence, that is $W^b_{\ell j} := w^{(j)}_{k^b}(Y^{(\ell)}_{k^b}; \mu^{0\to b})$, where $w^{(j)}_{k^b}$ is defined in (18). The weights are computed *incrementally* using the parent weights $W^a_{\ell j}$ through

$$\bar{W}^b_{\ell j} = W^a_{\ell j} \cdot U_{\ell j}, \text{ where } U_{\ell j} := \prod_{k=k^a}^{k^b} q(Y^{(\ell)}_k | X^{(j)}_k; \mu^{a\to b}_k), \tag{27a}$$

$$W^b_{\ell j} = \frac{\bar{W}^b_{\ell j}}{\sum_{j=1}^{L} \bar{W}^b_{\ell j}}. \tag{27b}$$

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

$$\hat{J}^b := \hat{J}_{N\mid k^b}(\mu^{0\to b}) = \sum_{\ell=1}^{L} \left\| \varphi(X^{(\ell)}_N) - \sum_{j=1}^{L} W^b_{\ell j} \varphi(X^{(j)}_N) \right\|^2. \tag{28}$$

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 $\mu^{0\to b}$, that is, 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, that is, $\eta^* = \eta^b$. The updated optimal vehicle trajectory can be backtracked from η^b to the root η^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. $\eta^b = \textbf{Sample}()$
2. $\eta^a \sim \mathcal{U}(\mathcal{T}^{\to b})$
3. $\mu^{a\to b} = \textbf{Connect}(\eta^a, \eta^b)$
4. $\mathcal{T} = \mathcal{T} \cup \{\eta^b\}$; $\rho^b = a$
5. compute W^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, that is, $\mathcal{T} = \{\eta^0\}$, and $\eta^* = \eta^0$.

5.1.1. 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 $\mu^{a\to b}$ from parent η^a to child node η^b rather than the complete trajectory $\mu^{0\to b}$. This is 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 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 an $O(\log(n))$ saving factor on average.

5.2. Example: simple vehicle

Consider a 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 Section 5.1 takes the form:

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

It begins by checking whether the trajectory to be computed must have a fixed final time k^b. When the final time epoch k^b is set to ∞ (line 1) then any k^b such that $k^a < k^b \leq N$ is allowed. This occurs when η^b is a uniform sample[2] in which case the trajectory $\mu^{a\to b}$ is generated using the 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 any obstacles then the connection fails (line 6).

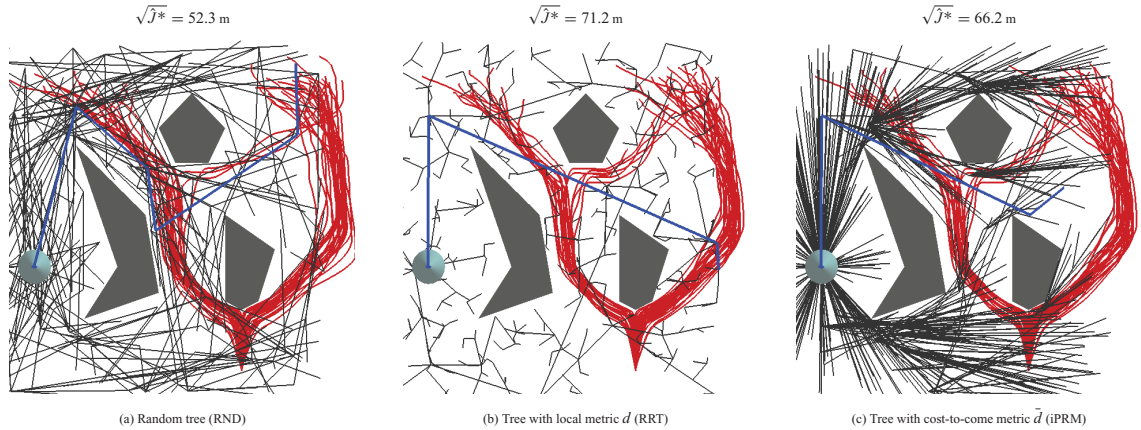


Fig. 3. Three types of search trees used to explore the vehicle trajectory space corresponding to the scenario in Section 2. The vehicle has simple dynamics and a circular sensing radius shown as a disk at its starting state. A subset of the target paths $X_{0:N}^{(1:L)}$ are shown diffusing from the bottom right to the top of the environment. The trees are: (a) a random tree (RND) constructed using **Expand** (Section 5.1); (b) a rapidly exploring random tree (RRT) using the nearest-neighbor metric d (29); (c) an incremental tree-based probabilistic road map (iPRM) expanded based on the cost-to-come distance $\bar{d}$ (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 m) relative to the environment size (200×200 m).

The optimal estimation algorithm is tested in a polygonal obstacle environment mimicking the scenario in Section 2. A tree built after calling **Expand** 500 times is shown in Figure 3(a). It takes a few milliseconds of computation to generate such a tree. In practice, a tree will contain tens of thousands of vertices. Figure 4 shows a few frames of the resulting motion along an optimal trajectory obtained by a denser tree with 10,000 nodes which took 5 s to compute. More detailed computational studies are performed in Section 7.

5.3. Other expansion methods

There are alternative methods of constructing 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-*distance metric* $d : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$.[3] 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

$$d(\eta^a, \eta^b) = k^b - k^a. \tag{29}$$

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 though the tree would approximate all states in the domain, most of the paths will be jagged and circuitous. For instance, Figure 3(b) shows such a tree computed for a scenario mimicking the setup of Figure 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 $c^0 = 0$. The nearest neighbor is then chosen according to a modified distance $\bar{d}$ defined by

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

A tree based on $\bar{d}$ will contain 'straighter' trajectories, see Figure 3(c), which take a minimum time to reach the reachable points in the state space. Such a tree is termed an *incremental probabilistic road map* (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 η^b is given by

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

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, that is, 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 paper:

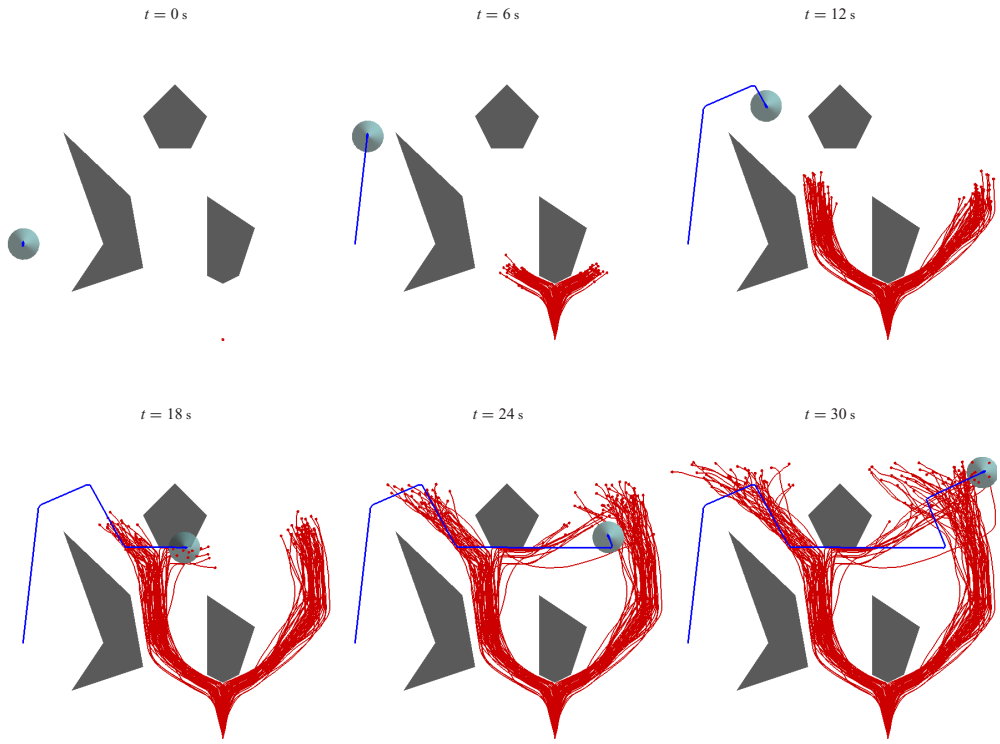


Fig. 4. The optimal vehicle path computed using algorithm **Expand** (Section 5.1) with $L = 100$ particles, time horizon $T = 30$ s, and time step $\tau = 250$ ms. The consecutive frames show the evolution of the sampled target trajectories $X_{0:N}^{(1:L)}$ and the vehicle trajectory $\mu_{0:N}^*$. The computed cost is $\sqrt{\hat{J}^*} = 43.9$ m.

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 η^0 to a uniformly sampled node $\eta^a \in \mathcal{N}$ reduces exponentially in n. More formally,*

$$P(\nexists \mu^{0\to a}) < \ell(1-c)^n,$$

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), for example, 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 ℓ), then adding more nodes would exponentially increase the probability of finding the path. The precise technical conditions that render Lemma (5.1) applicable to the scenario of this paper are developed in Hsu et al. (2002) and Frazzoli et al. (2002).

6. Variance-reduction techniques

The advantage of constructing the exploration tree described in Section 5 is that it asymptotically reaches arbitrarily close to any state in the state space $\mathcal{M}$ assuming that it is bounded. To guarantee that formally (as described in Section 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 (Rubinstein and Kroese 2008) as well as genetic computation, we propose three techniques that retain the probabilistic completeness of random trees but at the same time drastically speed up 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, for example, 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 $\mu_{0:N}^*$, that is, with lowest uncertainty cost J, is likely to pass close to states

with a high observation likelihood. Thus, a sample μ_k^{sample} is chosen so that

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

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

It is also possible to sample μ 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 Section 5.1 is specified as follows. It samples a state μ in two ways: 1) based on the utility (32) and 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. $\quad k \sim \mathcal{U}(\{0, \ldots, N\})$; $\ell \sim \mathcal{U}(\{1, \ldots, L\})$
3. $\quad \mu = \arg\max_{\mu'} q(\, Y_k^\ell | X_k^\ell; \mu')$, where $Y_k^\ell \sim q(\,\cdot | X_k^\ell; \mu')$
4. otherwise
5. $\quad k = \infty$
6. $\quad \mu \sim \mathcal{U}(\mathcal{M})$
7. repeat Sample if $h_c(\,\mu) < 0$
8. return $\eta = (\, k, \mu, \ldots\,)$

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 η^a and η^b. Whenever $k^b = \infty$, **Connect** is allowed to produce a trajectory $\mu^{a\to b}$ with any final time $k^b \le N$ that it chooses. In such cases **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 that arrives at μ^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 η^a at random. Then n_r other existing nodes are selected from the tree according to a 'fitness' function. Each of these nodes, denoted $\eta^b \in \mathcal{T} \backslash \{\eta^a \cup \eta^0\}$, are then disconnected from their current parents η^{ρ^b} and connected to η^a instead, as long as this switch lowers the resulting uncertainty cost of the subtree attached at η^b (see Figure 6).

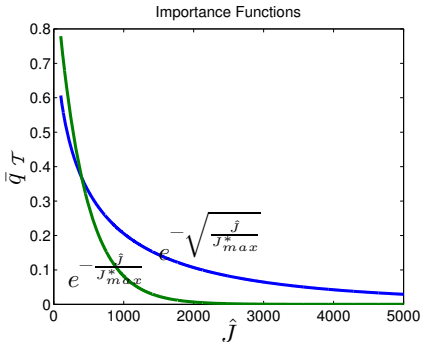


Fig. 5. Two importance density functions $\bar{q}_{\mathcal{T}}$ used to sample nodes during shuffling (with $J^*_{\max} = 400$). The function with the 'fatter' tail [defined by (33)] is the proper choice to guarantee exploration of the state space.

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

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

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 is achieved by a distribution with a fat tail as shown in Figure 5.

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

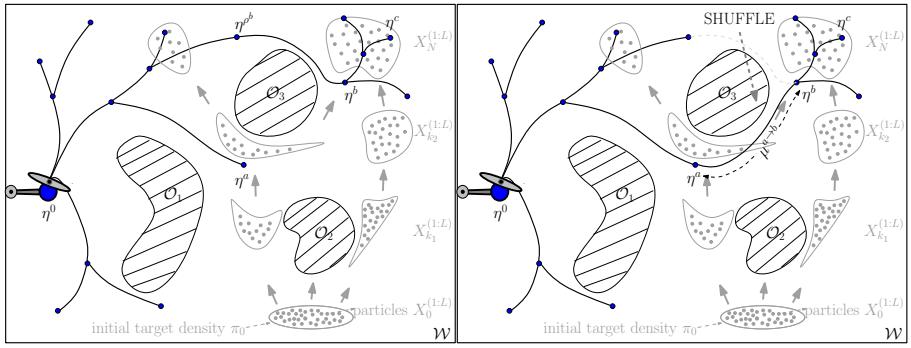
$$\mu^{a\to b\to c} := \mu^{a\to b} \cup \mu^{b\to c}.$$

More precisely, a *shuffle*, that is, a parent switch $\rho^b = a$, occurs in two cases (see also Figure 6). The obvious case is when the current optimal uncertainty cost $\hat{J}^*$ can be improved by a trajectory $\mu^{0\to a\to b\to c}$ 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

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

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

6.2.1. Computational savings The step (34) requires the computation of the uncertainty cost of all trajectories $\mu^{0\to a\to b\to c}$ obtained from the original $\mu^{0\to\rho^b\to b\to c}$ by replacing the segment $\mu^{\rho^b\to b}$ with $\mu^{a\to b}$. Since all trajectories in the subtree at η^b are affected this could be an expensive operation. In addition, it seemingly requires that the



(a) Current tree showing a path $\mu^{0\to b\to c}$ (b) Shuffled tree by deleting $\mu^{\rho^b\to b}$ and adding $\mu^{a\to b}$

Fig. 6. 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 probability inversely proportional to its current uncertainty cost $\hat{J}^b$; (b) η^b is disconnected from its parent η^{ρ^b} and connected to η^a after checking that the uncertainty cost of the newly formed trajectory $\mu^{0\to a\to b\to c}$ either improves the global optimum J^* or on average improves the costs in the subtree $\mathcal{T}^b$ [see (34)].

densities (27) at node η^a be re-propagated along the complete and potentially long new trajectory segment $\mu^{a\to b\to c}$. In the SIS framework though it is not necessary to perform the whole propagation. In particular, only the incremental weight update U_{ij} along the new segment $\mu^{a\to b}$ 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

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

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

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

Shuffle
1. choose η^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, Section 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 a tree by $\mathcal{L}_{\mathcal{T}} \subset \mathcal{T}$, that is,

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

Nodes are removed sequentially from the 'bottom' of the tree, starting with leaf nodes. The procedure is summarized as

Prune
1. for $i = 1, \ldots, n_p$
2. $\mathcal{L}' = \mathcal{L}_{\mathcal{T}} \backslash \{\eta^*, \eta^0\}$
3. $\eta \sim 1 - q_{\mathcal{L}'}(\cdot)$
4. $\mathcal{T} = \mathcal{T} \backslash \{\eta\}$

The probability of choosing nodes for pruning is inversely proportional to their fitness density (line 3). Empirically, as shown in Section 7, pruning is an effective strategy [again in the spirit of importance (re)sampling] for obtaining improved solutions more quickly. 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, that is, $n_p = n/5$.

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, that is, to reach the optimum as the number of algorithm iterations tends to infinity.

The random expansion algorithm (RND) (Section 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 optimum as the number of iterations increases. Yet, this might be an infinitely slow process in practice, a fact also confirmed by simulations in Section 7.

Utility-based importance sampling (Section 6.1) can provide a good solution more quickly 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 a particular problem structure

and regularity conditions. Currently, the advantage of this proposed technique is observed empirically for specific problems.

The idea of shuffling (Section 6.2) is to approximate an optimum not by generating completely new trajectories (as RND does) but instead by executing local modifications to the existing tree structure. Assume that the optimal trajectory, which the algorithm aims to compute, consists of $m+1$ nodes $\eta^0, \bar{\eta}^1, \ldots, \bar{\eta}^m$. 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 $\bar{\eta}^i$, $i = 1, \ldots, m$. 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, that is 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 (Section 6.3) is, then, to remove less promising nodes in order to reduce the search space. In essence, the size of the solution 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 were tested through multiple simulation runs in the simulated scenario defined in Section 2 with the tree node connection defined in Section 5.2. Four algorithms were developed and analyzed in order to compare the proposed baseline algorithm and variance-reduction techniques. The algorithms are:

- RND: baseline random expansion algorithm (Section 5.1)
- RND+UTIL: RND augmented with utility-based sampling (Section 6.1)
- RND+UTIL+SHUFFLE: with the addition of a shuffling step (Section 6.2)
- RND+UTIL+SHUFFLE+PRUNE: the final algorithm including pruning (Section 6.3).

Figure 7 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 Figure 9. The simulations shows that a random search tree (RND) is better than a standard motion-planning tree for obtaining convergence to an optimal trajectory. Yet 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 s. Several snapshots of the dynamic tree and the resulting optimal trajectory as the algorithm progresses are shown in Figure 8. Note that all reported results are based on a global open-loop optimization with simulated future measurements rather than in a receding horizon fashion.

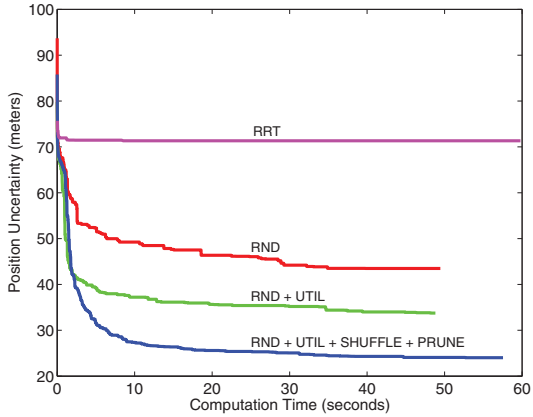


Fig. 7. Comparison of several tree-search algorithms. The y-axis plots the square root of the total variance $\sqrt{J*}$, which can be interpreted as the combined error for the two-position coordinates. A standard RRT algorithm is not suitable for optimal planning since it quickly converges to and remains at a low-quality solution. The algorithm RND converges asymptotically but the rate decreases with computation time. Utility-based sampling (RND+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 m.

Table 3. Parameters for the complete algorithm RND+UTIL+SHUFFLE+PRUNE

Parameter	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	$\dim(\mathcal{T})/5$

Finally, it should be noted that the complete algorithm requires several parameters, which must be selected in advance. These are listed in Table 3.

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.

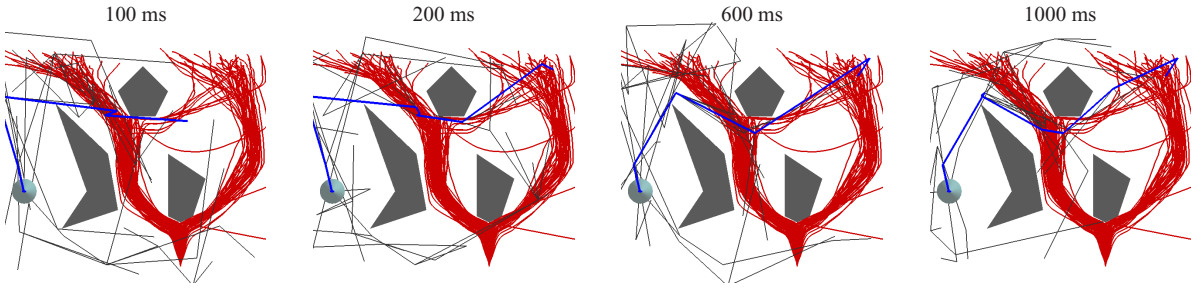


Fig. 8. Several frames showing the current tree and optimal trajectory computed by algorithm RND+UTIL+SHUFFLE+PRUNE at different computation times.

7.2. A helicopter search scenario

Consider a small autonomous helicopter as depicted in Figure 10 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 $R \in \mathrm{SO}(3)$ where $\mathrm{SO}(3)$ denotes the space of right-handed coordinate frames described by three orthogonal vectors (i.e. by a 3×3 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 $\mathbb{J} = \mathrm{diag}(J_1, J_2, J_3)$.

The vehicle is controlled through a *collective* u_c (lift produced by the main rotor) and a *yaw* u_ψ (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* γ_p and sideways through a *roll* γ_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 $\mathcal{M} = SO(3) \times \mathbb{R}^3 \times \mathbb{R}^6 \times \mathbb{R}^2$ with $\mu = ((R, p), (\omega, v), \gamma)$.

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} R\,\widehat{\omega} \\ R\,v \end{bmatrix}, \tag{36}$$

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

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

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

while the control matrix is defined as

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

The local motion-planning method corresponding to **Connect** is based on sequencing the precomputed motion primitives that satisfy the dynamics, (36) and (37). This is accomplished using a maneuver automaton described in Appendix 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—$\mathrm{SE}(2) \times \mathbb{R}^1$. A trajectory consisting of a given sequence of a minimum number of primitives can then be computed instantly in closed form through inverse kinematics. Figure 10(b) 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), which computes the closest distance between arbitrary polyhedra and is used to implement the function **prox** defined in (5).

The algorithm is tested using a scenario similar to that in Section 2, now extended to 3-D. The helicopter is not permitted to fly above obstacles. Figure 11 shows the resulting helicopter trajectory [see also the video in Extension 1 (Section A)], a view of the constructed road map, and a close-up along the optimal path within the road map.

8. Conclusion

This paper deals with optimal estimation for systems with non-linear dynamics subject to non-convex constraints. The approach is based on a random enumeration of trajectories generated from a tree that compactly approximates the reachable space and efficiently propagates probability distributions through recursion. The randomly sampled tree nodes approach any reachable state with exponentially (for the number of iterations) high probability and, therefore, encode a versatile road map 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

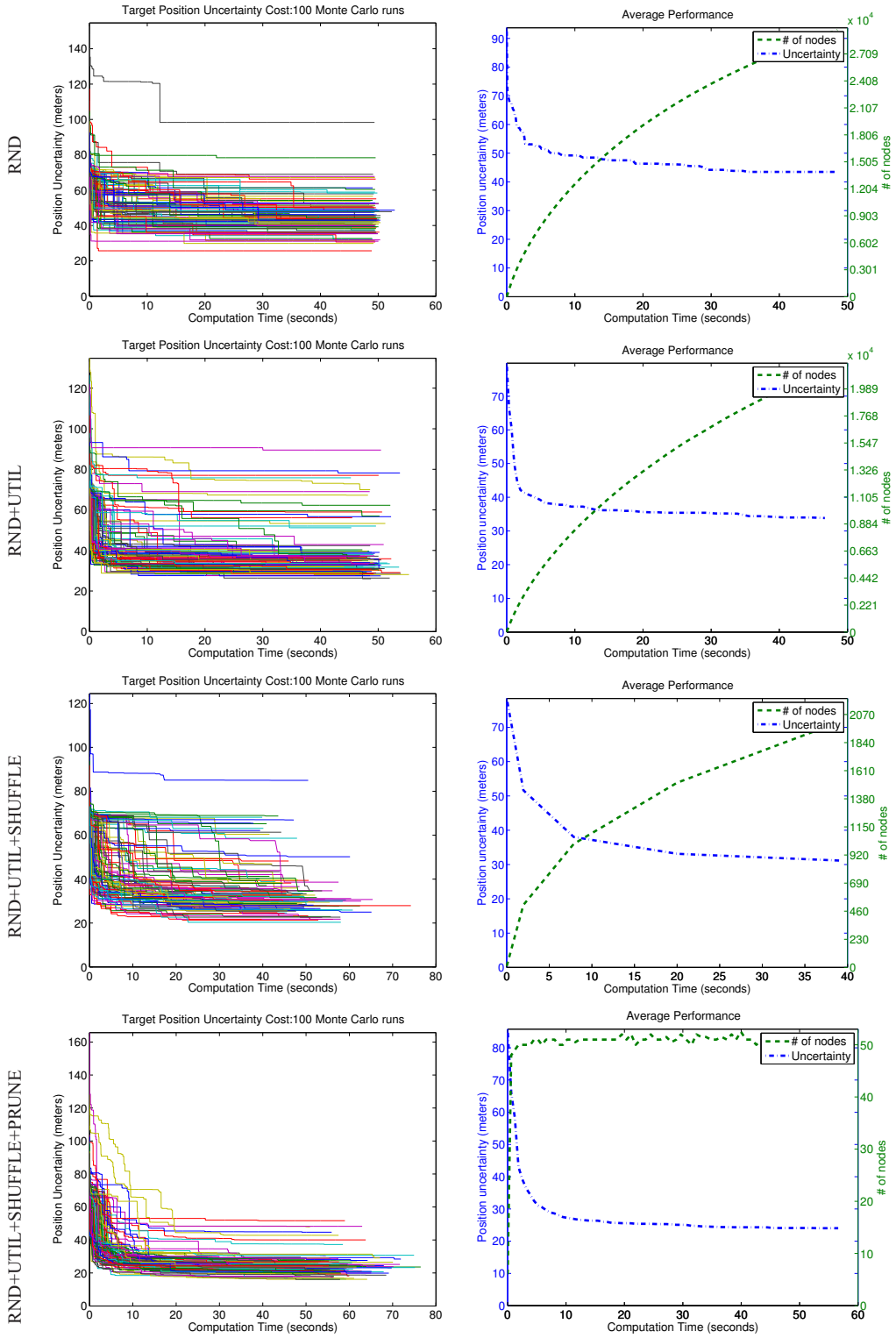


Fig. 9. Monte Carlo analysis of the the proposed techniques applied to the scenario in Section 2. The performance metrics are the resulting optimal uncertainty cost $\sqrt{\hat{J}^*}$ (which can be regarded roughly as the standard deviation of the distance to the true value) and the corresponding number of tree nodes. These two metrics are expressed in terms of the required computation time (shown on the x-axis). The left plots show all 100 Monte Carlo runs while the right plots shows the averaged results.

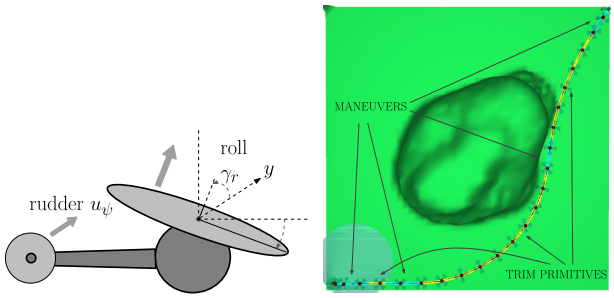


Fig. 10. (a) Simplified helicopter model used in our tests. (b) Example of a computed sequence of four maneuvers and three trim primitives, connecting two zero-velocity states in the corners of the workspace and avoiding an obstacle in the center. The trim velocities satisfy the invariance conditions defined in Section B.1 while the maneuvers are computed as outlined in Section B.2.

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 paper provides a simple particle-based algorithm applicable to general types of dynamics and uncertainty models, which is easy to implement and performs well in practice.

Notes

1. 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.
2. A uniform sample is sampled uniformly over the state space; additional sampling choices will be given in Section 6.1.
3. d is not required to be a true distance metric, i.e. it does not need to be symmetric or necessarily satisfy the triangle inequality (LaValle 2006).

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of interest statement

None declared.

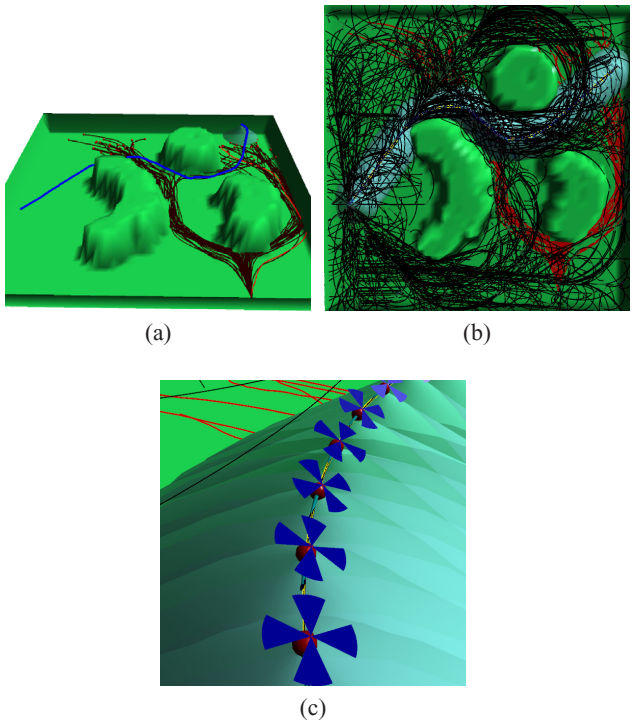


Fig. 11. The algorithm applied to the helicopter example described in Section 7.2 showing (a) the optimal helicopter trajectory; (b) the constructed road map and sensor footprint along a path; and (c) the close-up view along an edge of the road map.

Acknowledgement

The authors thank the reviewers for their comprehensive review and advice for improving this paper.

References

Bethke B, Bertuccelli LF, and How JP (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-PL (2007) Optimal path planning using cross-entropy method. In: *2006 9th International Conference on Information Fusion*, 1–8.

Choset H, Lynch KM, Hutchinson S, Kantor GA, Burgard W, Kavraki LE and Thrun S (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

Chung TH and Burdick JW (2007) A decision-making framework for control strategies in probabilistic search. In: *Intl. Conference on Robotics and Automation*. ICRA, 4386–4393.

Clarke FH, Ledyaev YS, Stern RJ and Wolenski PR (1998) *Nonsmooth Analysis and Control Theory*. Springer.

Cole DT, Goktogan AH and Sukkarieh S (2008) The demonstration of a cooperative control architecture for UAV teams. In:

Experimental Robotics. Berlin, Heidelberg: Springer, vol. 39, 501–510.

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*, vol. 3, 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 (eds) (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*, 1843–1849.

Frazzoli E, Dahleh MA and Feron E (2002) Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics* 25: 116–129.

Frazzoli E, Dahleh MA and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21: 1077–1091.

Geyer C (2008) Active target search from UAVs in urban environments. In: *Proc. IEEE International Conference on Robotics and Automation*. ICRA, 2366–2371.

Gottschalk S, Lin MC 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, vol. 1, 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, 1814–1820.

Hocaoglu C and Sanderson AC (2001) Planning multiple paths with evolutionary speciation. *IEEE Transactions on Evolutionary Computing* 5: 169–191.

Hoffmann GM and Tomlin CJ (2010) Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control* 55(1): 32–47.

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: 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: 229–242.

Langdon WB and Poli R (2001) *Foundations of Genetic Programming*. Springer.

Latombe J-C (1991) *Robot Motion Planning*. Kluwer Academic Press.

LaValle SM (2006) *Planning Algorithms*. Cambridge, UK: Cambridge University Press.

Lavis B, Furukawa T and Whyte HFD (2008) Dynamic space reconfiguration for Bayesian search and tracking with moving targets. *Auton. Robot* 24: 387–399.

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: 93–103.

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 JD (2003a) A comparison of decision making criteria and optimization methods for active robotic sensing. In: *Lecture Notes in Computer Science*, vol. LNCS 2542, 316–324.

Mihaylova L, Schutter JD and Bruyninckx H (2003b) A multisine approach for trajectory optimization based on information gain. *Robotics and Autonomous Systems* 43: 231–243.

Paris S and Le Cadre J-P (2002) Planning for terrain-aided navigation. In: *Proc. 5th Int Information Fusion Conf.*, vol. 2, 1007–1014.

Powell W (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics.

Robert CP and Casella G (2004) *Monte Carlo Statistical Methods*. Springer.

Rubinstein RY and Kroese DP (2008) *Simulation and the Monte Carlo Method*. Wiley.

Rubinstein RY and Kroese DP (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, 661–666.

Singh SS, Kantas N, Vo B-N, Doucet A and Evans RJ (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: 35–42.

Tremois O and Le Cadre J-P (1999) Optimal observer trajectory in bearings-only tracking for maneuvering sources. *IEEE Proceedings – Radar, Sonar and Navigation* 146: 31–39.

Vaidyanathan R, Hocaoglu C, Prince TS and Quinn RD (2001) Evolutionary path planning for autonomous air vehicles using multi-resolution path representation. In: *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 1, 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: 18–28.

Appendices

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, that is, one that 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 by Frazzoli et al. (2005), is suitable for systems such as UAVs or ground robots if one ignores pose-dependent external forces, such as varying wind or ground friction that is a function of position.

Let the vehicle rotation be described by its roll ϕ, pitch θ, and yaw ψ. 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

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

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

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

where f_u is the control force, corresponding to $F(\gamma)u$ in (37), while $f_{\mathrm{ext}} = (0, 0, 0, 0, 0, -9.81\ \mathrm{m}) \in \mathfrak{se}(3)^*$ is the gravity force. Since gravity is the only configuration-dependent term in the dynamics and is invariant for translations and rotations about 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 for the trims and maneuvers used is described next. An example sequence is shown in Figure 10.

B.1. Trim primitives

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

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

since f_{ext} is G-invariant.

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

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

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, 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 $\pi : X \to X \backslash G$ 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:

Compute: $T;\ x : [0, T] \to X;\ u : [0, T] \to U$

minimizing: $J(x, u, T) = \int_0^T 1 + \lambda \|u(t)\|^2 dt$

subject to: $\pi(x(0)) = x_1, \pi(x(T)) = x_2,$
dynamics equations (36) and (37)
for all $t \in [0, T]$.

The parameter λ 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 calculate this by using a 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 non-linear state spaces and symmetries. A geometric structure preserving the optimizer, developed by (Kobilarov 2008, Section 2.7), is used to perform the computations. The computations are performed off-line with the resulting optimal maneuvers assembled in a library offering an instant look-up during run-time.