Sampling-based Planning for Sensor Scheduling in Constrained Environments

Marin Kobilarov (Johns Hopkins University), Gaurav S. Sukhatme (University of Southern California)
Jerrold Marsden (California Institute of Technology)

***Abstract*—This paper 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.**

Note: the paper is a condensed version of the journal article [1].

I. Introduction

Consider a mobile sensor (a *vehicle*) estimating the state of an observable (a *target*) with known stochastic 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

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

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

where Ω_k and V_k are 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 $0\leq i<j\leq N$, is denoted by $x_{i:j}$, i.e. $x_{i:j}:=\{x_i,x_{i+1},...,x_{j-1},x_j\}$.

[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.

The vehicle trajectory $\mu_{0:N}$ is subject to dynamical constraints, e.g. arising from discretized Euler-Lagrange equations of motion, expressed through

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

where u_k denotes the vehicle control input at time k. In this work we assume that a local boundary value method is available to compute the the required controls in order to generate a trajectory to a given state. For instance, for dynamical systems which are feedback linearizable [2] it is possible to use spline-based interpolation techniques while for systems with symmetries such as the helicopter example described in §VIII-B one can employ planning with primitives and inverse kinematics.

In addition the vehicle must avoid obstacles and is subject to velocity and actuator bounds, jointly encoded through

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

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*.

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 $y_{1:k}$.

A. 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, 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 through

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

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 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 covariance determinant, this metric is chosen since its value can be interpreted in meaningful units (e.g. see 3). 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 [4].

The optimization (4) corresponds to the *optimal sensor scheduling* problem [5, 6] which is of central importance in the target tracking community. It is also highly relevant to the problem of *active sensing* studied in robotics [7, 8, 3] where the vehicle is estimating its own state [9, 10] and in some cases refining its knowledge about the environment [11, 12].

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 [9, 13, 6] is not suitable unless a good starting guess is chosen since the constraints impose many local minima. In addition, special differentiation [14] 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 [15]. 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 [16] 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) *Biased sampling*: node sampling based on expected utility of improving the target estimate in order to focus tree exploration into more "promising" parts of the state space.

2) *Shuffling*: random modification of 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) *Pruning*: removal of existing nodes probabilistically according to their performance.

These proposed methods result in a significant computational speed-up compared to a random baseline algorithm. The proposed combination of techniques is motivated by the need to address the exploration-exploitation trade-off paradigm (see e.g. 17) with computational efficiency by dynamically adjusting the search space. 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.

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 [18] 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 19). Standard techniques, i.e. penalty functions or infeasible path rejection, employed in works such as [20, 21, 22, 23] 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.

II. An Example Scenario

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 $\mathcal{W}=\mathbb{R}^3$ [24]. The workspace contains a number of *obstacles* denoted by $\mathcal{O}_1,...,\mathcal{O}_{n_o}\subset\mathcal{W}$ 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

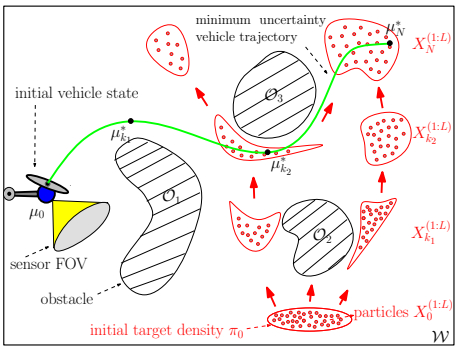


Fig. 1. A scenario with a vehicle (depicted as a small helicopter) at state $\mu_0 \in \mathcal{M}$ and a target with initial distribution π_0 diffusing north. Both target and vehicle avoid obstacles denoted $\mathcal{O}_i$. The set of possible target motions is approximated by L sampled trajectories $X_{0:N}^{(\ell)}$ 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 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.

in (3) is then to avoid obstacles, generally expressed as

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

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 §VIII-B.

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 τ 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 Ω with standard deviations σ_x and σ_y. The model is

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

where τ 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 corresponds to the function f defined in (1a).

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

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

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

$$g^*((r,v);(r',v')) = \begin{bmatrix} \|r-r'\| \\ \arctan((r_y-r'_y),(r_x-r'_x)) \end{bmatrix}, \quad (7)$$

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

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

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 $[0,1]$.

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).

III. Problem Formulation

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

$$X_0 \sim \pi_0, \quad (8a)$$
$$X_k \sim p(\cdot|X_{k-1}), \qquad k>0 \quad (8b)$$
$$Y_k \sim q(\cdot|X_k;\mu_k), \qquad k>0 \quad (8c)$$

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. 25) according to

$$\pi_k(x|y_{1:k};\mu_{1:k}) = \frac{q(y_k|x;\mu_k)\int p(x|x')\pi_{k-1}(x'|y_{1:k-1};\mu_{1:k-1})dx'}{\int q(y_k|x;\mu_k)\int p(x|x')\pi_{k-1}(x'|y_{1:k-1};\mu_{1:k-1})dx'dx}. \quad (9)$$

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

$$\pi_{k+i|k}(x|y_{1:k};\mu_{1:k}) = \int p(x|x')\pi_{k+i-1|k}(x'|y_{1:k};\mu_{1:k})dx', \quad (10)$$

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

$$\begin{aligned}\Phi_{N|k}(y_{1:k};\mu_{1:k}) &:= \mathbb{E}[\varphi(X_N) \mid y_{1:k};\mu_{1:k}] \\ &= \int \varphi(x_N)\pi_{N|k}(x|y_{1:k};\mu_{1:k})dx.\end{aligned} \quad (11)$$

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 according to

$$\begin{aligned}J_{N|k}(\mu_{1:k}) &= \mathbb{E}\left[\|\varphi(X_N)-\Phi_{N|k}(Y_{1:k};\mu_{1:k})\|^2\right] \\ &= \int \|\varphi(x_N)-\Phi_{N|k}(y_{1:k};\mu_{1:k})\|^2 p(x_{0:N},y_{1:k}|\mu_{1:k})dx_{0:N}dy_{1:k}.\end{aligned} \quad (12)$$

The expectation over states and measurements in (12) is taken with respect to the density $p(x_{0:N},y_{1:k}|\mu_{1:k})$ which, for Markov models in the form (8), can be decomposed (see e.g. [25]) as

$$p(x_{0:N},y_{1:k}|\mu_{1:k}) = \pi_0(x_0)\prod_{i=1}^{N}p(x_i|x_{i-1})\prod_{i=1}^{k}q(y_i|x_i;\mu_i). \quad (13)$$

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

$$\mu^*_{1:N} = \operatorname*{argmin}_{\mu_{1:N}} J(\mu_{1:N}), \quad (14)$$

subject to the dynamics and constraints.

IV. Sampling-based Approximation

The filtering densities (9) and (10) generally cannot be computed in closed form since they are based on nonlinear/non-Gaussian models. Therefore, following [6], we employ particle-based approximation using L delta distributions, placed at state *samples* $X_k^{(j)}\in\mathcal{X}$ with positive weight functions $w_k^{(j)}:\mathcal{Y}^k\times\mathcal{M}^k\to\mathbb{R}_+$, i.e.

$$\pi_k(x|y_{1:k};\mu_{1:k}) \approx \hat{\pi}_k(x|y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L} w_k^{(j)}(y_{1:k};\mu_{1:k})\delta_{X_k^{(j)}}(x), \quad (15)$$

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 (8a) and target motion model (8b) and to compute the weights, for given measurements $y_{1:k}$ obtained at vehicle states $\mu_{1:k}$, according to

$$\bar{w}_k^{(j)}(y_{1:k};\mu_{1:k}) := \prod_{i=1}^{k} q(y_i|X_i^{(j)};\mu_i), \quad (16)$$

$$w_k^{(j)} = \frac{\bar{w}_k^{(j)}}{\sum_{\ell=1}^{L}\bar{w}_k^{(\ell)}}, \quad (17)$$

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 $\pi_0(x_0)\Pi_{i=1}^{k}p(x_i|x_{i-1})$. Note that while more sophisticated sampling methods have been developed, e.g. that additionally account for measurements to reduce variance [26, 25], 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 (10) is approximated simply according to

$$\pi_{k+i|k}(x|y_{1:k};\mu_{1:k}) \approx \hat{\pi}_{k+i|k}(x|y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L} w_k^{(j)}(y_{1:k};\mu_{1:k})\delta_{X_{k+i}^{(j)}}(x). \quad (18)$$

The estimate (11) is then approximated by

$$\Phi_{N|k}(y_{1:k};\mu_{1:k}) \approx \hat{\Phi}_{N|k}(y_{1:k};\mu_{1:k}) := \sum_{j=1}^{L}\varphi(X_N^{(j)})\hat{\pi}_{N|k}(X_N^{(j)}|y_{1:k};\mu_{1:k}). \quad (19)$$

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 (19) degrades since the probability mass becomes concentrated in a decreasing number of particles [25]. A standard remedy is to include a resampling step [26] 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 (15) and estimate (19) 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 (12), i.e. 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 (15). Measurement sequences $Y_{1:k}^{(\ell)}$ are then sampled by drawing $Y_i^{(\ell)}\sim q(\cdot|X_i^{(\ell)};\mu_i)$ for all $i=1,...,k$. As long as the densities (8) can be directly sampled from, which is valid for common models used in robotics (e.g. 7), then the approximation simplifies to the Monte Carlo or the stochastic counterpart, i.e.

$$J_{N|k}(\mu_{1:k})\approx\hat{J}_{N|k}(\mu_{1:k}):=\frac{1}{L}\sum_{\ell=1}^{L}\|\varphi(X_N^{(\ell)})-\hat{\Phi}_{N|k}(Y_{1:k}^{(\ell)};\mu_{1:k})\|^2, \tag{20}$$

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 (19) and cost (20), i.e. it will solve

$$\hat{\mu}_{0:N}^{*}=\underset{\mu_{0:N}}{\operatorname{argmin}}\hat{J}(\mu_{0:N}). \tag{21}$$

In this sense only an approximate solution will be obtained. Yet, by the law of large numbers [27] $\hat{\mu}_{0:N}^{*}$ will approach the true solution $\mu_{0:N}^{*}$ by increasing the number of simulations L.

V. Random Tree Optimization

We employ tree-based search based on *sampling* nodes from the original continuous space $\mathcal{M}$ and connecting them with 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) [15] and the *probabilistic roadmap* (PRM) [28]. 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 §VI introduces the variance reduction techniques that complete the overall approach.

A. 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 μ; 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 ρ. Nodes and their sub-elements are indexed by superscript, i.e. 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 $W_{\ell j}^0=1/L$ for all $\ell,j=1,...,L$. 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 $k^a\leq k\leq k^b$.

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 $W_{\ell j}^b:=w_{k^b}^{(j)}(Y_{1:k^b}^{(\ell)};\mu^{0\to b})$
$\hat{J}^b$	scalar >0	uncertainty cost $\hat{J}^b:=\hat{J}_{N|k^b}(\mu^{0\to b})$
ρ^b	integer ≥ 0	parent node index

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

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 $\mu^{0\to a}$. 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 [15] 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 $\mu^{a\to b}$ is produced, otherwise the connection fails. Such a controller is abstractly represented by the function **Connect**, i.e.

$$\textbf{Connect}(\eta^a,\eta^b)\Rightarrow\begin{cases}\mu^{a\to b}, & \text{if path found}\\ \emptyset, & \text{otherwise.}\end{cases} \tag{22}$$

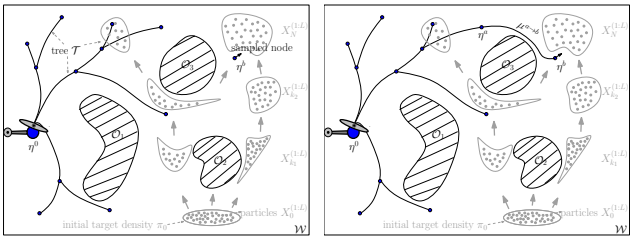
The tree is constructed by sampling and connecting nodes. Assume that a function **Sample** is available which returns a new node, i.e.

$$\eta^b=\textbf{Sample}(). \tag{23}$$

The default choice is to sample (state,time) pairs (μ,k) uniformly from $\mathcal{M}\times\{1,...,N\}$ and discard samples that violate the constraints (3), e.g. that 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, i.e.

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

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 $\mu^{a\to b}$, i.e. $\rho^b=a$. Fig. 3 illustrates the construction.



a) tree $\mathcal{T}$ and a sampled node η^b b) connecting to η^b

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

After a new node η^b is added to the tree, the target filtering density $\hat{\pi}$ (18) is propagated along the newly added trajectory segment $\mu^{a\to b}$ for all sampled target paths $X_{k^a:k^b}^{(\ell)}$

by simulating measurements

$$Y_k^{(\ell)} \sim q(\cdot|X_k^{(\ell)};\mu_k^{a\to b}), \quad \text{for } k=k^a,...,k^b, \tag{24}$$

for all $\ell=1...L$. A row in the matrix W^b, i.e. W_ℓ^b for any $1\leq\ell\leq L$ corresponds to the resulting weights for each measurement sequence, i.e. $W_{\ell j}^b := w_{k^b}^{(j)}(Y_{k^b}^{(\ell)};\mu^{0\to b})$, where $w_{k^b}^{(j)}$ is defined in (16). The weights are computed *incrementally* using the parent weights $W_{\ell j}^a$ through

$$\bar{W}_{\ell j}^b = W_{\ell j}^a \cdot U_{\ell j}, \text{ for } U_{\ell j} := \prod_{k=k^a}^{k^b} q(Y_k^{(\ell)}|X_k^{(j)};\mu_k^{a\to b}), \tag{25a}$$

$$W_{\ell j}^b = \frac{\bar{W}_{\ell j}^b}{\sum_{j=1}^L \bar{W}_{\ell j}^b}. \tag{25b}$$

The error (20) of the complete trajectory $\mu^{0\to b}$, denoted by $\hat{J}^b$, becomes

$$\hat{J}^b := \hat{J}_{N|k^b}(\mu^{0\to b}) = \sum_{\ell=1}^L \|\varphi(X_N^{(\ell)}) - \sum_{j=1}^L W_{\ell j}^b \varphi(X_N^{(j)})\|^2. \tag{26}$$

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 $\mu^{0\to b}$, i.e. up to time k^b. If $\hat{J}^b < \hat{J}^*$, where $\hat{J}^b$ is computed using (26) 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 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) η^b=**Sample**()
2) $\eta^a\sim\mathcal{U}(\mathcal{T})$
3) $\mu^{a\to b}$=**Connect**(η^a,η^b)
4) $\mathcal{T}=\mathcal{T}\cup\{\eta^b\}$; $p^b=a$
5) Compute W^b and $\hat{J}^b$ using (25) and (26)
6) if $J^b<J^*$ 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 (26) 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 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 (25) 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.

B. Example: simple vehicle

Consider the simple vehicle with dynamics (6). The optimal estimation algorithm is tested in a polygonal obstacle environment mimicking the scenario in §II. 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 §VIII.

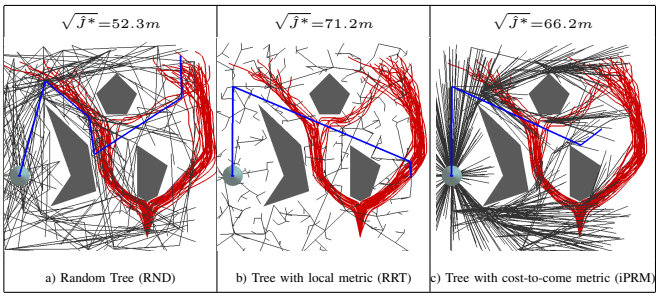


Fig. 4. Three types of search trees used to explore the vehicle trajectory space corresponding to the scenario in §II. The vehicle has simple dynamics and a circular sensing radius shown as a disk at its starting state. A subset of the target paths $X_{0:N}^{(1:L)}$ are shown diffusing from the bottom right to the top of the environment. The trees are: a) a random tree (RND) constructed using **Expand** (§V-A); b) a rapidly-exploring random tree (RRT) using nearest neighbor metric; c) an incremental tree-based probabilistic roadmap (iPRM) expanded based on cost-to-come distance. 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 (200×200 meters.).

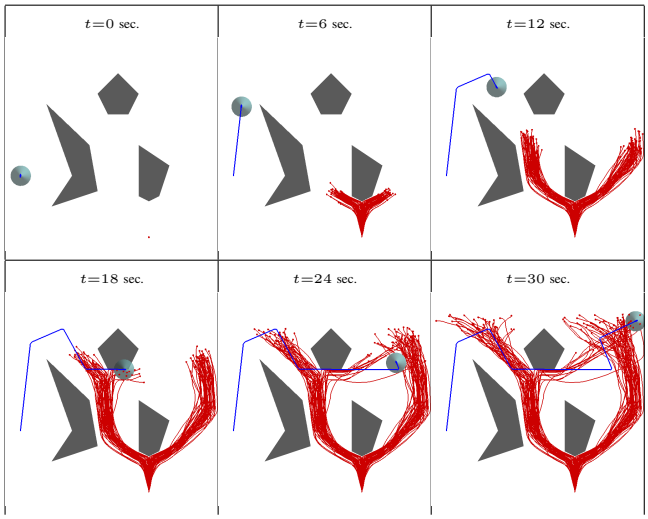


Fig. 5. The optimal vehicle path computed using algorithm **Expand** (§V-A) with L=100 particles, time horizon T=30 s., and time-step τ=250 ms. The consecutive frames show the evolution of the sampled target trajectories $X_{0:N}^{(1:L)}$ and the vehicle trajectory $\mu_{0:N}^*$. The computed cost is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

VI. Variance Reduction Techniques

A. 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. 29).

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 μ_k^{sample} is chosen so that

$$\mu_k^{\text{sample}}=\underset{\mu}{\text{argmax}} q(Y_k^\ell|X_k^\ell;\mu), \tag{27}$$

where $(X_k^{(\ell)},Y_k^{(\ell)})$ is a single particle selected by sampling ℓ uniformly from $\{1,...,L\}$. The optimal state μ in (27) is usually straightforward to compute. For instance, the optimal vehicle position in the example scenario from §II will coincide (on average) with the target position at X_k^ℓ so according to (27) one can simply set $\mu_k^{\text{sample}}=X_k^\ell$.

The function **Sample** introduced in §V-A is specified as follows. It samples a state μ in two ways: 1) based on the utility (27); 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) $k\sim\mathcal{U}(\{0,...,N\})$; $\ell\sim\mathcal{U}(\{1,...,L\})$;
3) $\mu=\text{argmax}_{\mu'} q(Y_k^\ell|X_k^\ell;\mu')$, 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 $\eta=(k,\mu,...)$

B. 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 Fig. 7).

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

$$q_\mathcal{T}(\eta^b)=\frac{\bar{q}_\mathcal{T}(\eta^b)}{\sum_{\eta\in\mathcal{T}}\bar{q}_\mathcal{T}(\eta)}, \text{ where } \bar{q}_\mathcal{T}(\eta^b)=e^{-\sqrt{\frac{\hat{J}^b}{J^*_{\text{max}}}}}, \tag{28}$$

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.

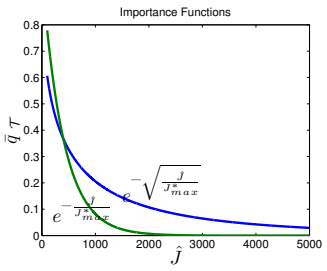


Fig. 6. Two importance density functions $\bar{q}_\mathcal{T}$ used to sample nodes during shuffling (with J^*_{max}=400). The function with "fatter" tail (defined in (28)) is the proper choice in order to guarantee exploration of the state space.

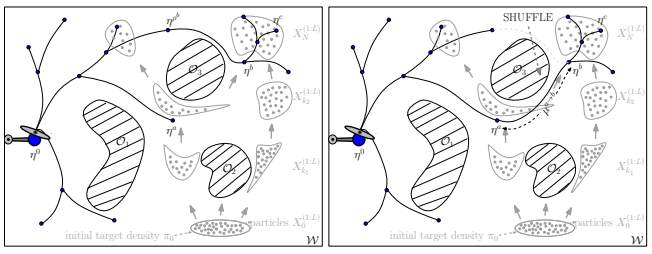
Let the *subtree* rooted at η^b be denoted $\mathcal{T}^b\subset\mathcal{T}$. Define the combined trajectory connecting node η^a to node η^b and node η^b to node η^c, denoted $\mu^{a\to b\to c}$, by

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

More precisely, a *shuffle*, 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 $\mu^{0\to a\to b\to c}$ 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

$$\text{if } \left\{\begin{array}{c}\min_{\eta^c\in\mathcal{T}^b}\hat{J}_{N|k^c}(\mu^{0\to a\to b\to c})<\hat{J}^*\\ \text{or}\\ \sum_{\eta^c\in\mathcal{T}^b}\left(\hat{J}_{N|k^c}(\mu^{0\to a\to b\to c})-\hat{J}^c\right)<0\end{array}\right\} \text{ then } \rho^b=a. \tag{29}$$

Note that $\hat{J}^c$ in (29) should be understood as the present cost in the unmodified tree, i.e. $\hat{J}^c:=\hat{J}_{N|k^c}(\mu^{0\to\rho^b\to b\to c})$.



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. 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) 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 (29)).

Computational Savings.: The step (29) 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 densities (25) 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 (25)) and then the weights at all subtree nodes $\eta^c{\in}\mathcal{T}^b$ are updated directly through the simple *weight rescaling* formula

$$\bar{W}_{ij}(\mu^{0\to a\to b\to c}){=}W^c_{ij}\frac{W^a_{ij}}{W^b_{ij}}U_{ij}, \tag{30}$$

where the weights W^a_{ij} are the existing weights at node η^a, resp. b and c. After computing the unnormalized weights (30) the cost $\hat{J}_{N|k^c}(\mu^{0\to a\to b\to c})$ used in the shuffling (29) is computed through (25b) and (26).

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

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 (29)
5) if $J^c{<}J^*$ 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 $n_r{=}\log(\dim(\mathcal{T}))$.

C. Randomized Pruning

Shuffling §VI-B 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) $\mathcal{L}'{=}\mathcal{L}_{\mathcal{T}}\backslash\{\eta^*,\eta^0\}$
3) $\eta{\sim}1{-}q_{\mathcal{L}'}(\cdot)$
4) $\mathcal{T}{=}\mathcal{T}\backslash\{\eta\}$

Nodes to be pruned are selected inversely proportional to their fitness density (line 3). Empirically, as shown in §VIII, pruning turns out 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. $n_p{=}n/5$.

VII. Computational Aspects

The proposed methods are expected to find only an approximate solution as a result of the sampling-based approximation of the target dynamics and the variance reduction techniques. While we do not formally study the algorithm convergence properties, it is expected that the number of iterations required for a near-optimal solution by the purely random algorithm described in §V is prohibitively expensive due to the size of the search space [30]. Employing variance-reduction methods drastically speeds up the search at the expense of loosing algorithmic completeness. More specifically, by biasing the search and pruning the tree heuristically we introduce an undesirable probability of failing to find a near-optimal solution. An important point that we have not considered is to quantify this probability precisely, i.e. in terms of probably-approximately-correct (PAC) analysis [31]. Despite these limitations, the numerical tests presented next suggest that the proposed algorithms are suitable for real-time implementation in a receding horizon fashion, by choosing a suitably short planning hirozon. More specifically, the current implementation requires several seconds of computation to reach an acceptable solution. This run-time can also be significantly lowered through parallelization.

VIII. Numerical Tests

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

A. Simple Vehicle

The methods are tested through multiple simulation runs in the simulated scenario defined in §II with tree node connection defined in §V-B. 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 (§V-A)
- RND+UTIL: RND augmented with utility-based sampling (§VI-A)
- RND+UTIL+SHUFFLE: with the addition of a shuffling step (§VI-B)
- RND+UTIL+SHUFFLE+PRUNE: the final algorithm including pruning (§VI-C)

Fig. 8 shows the resulting averaged results of the performance of each algorithm, including a comparison with a standard RRT expansion (see [1] for more detailed results). 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.

B. A Helicopter Search Scenario

Consider a small autonomous helicopter operating in a 3-D terrain. The vehicle is modeled as a single underactuated rigid body with position $p{\in}\mathbb{R}^3$ 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 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.

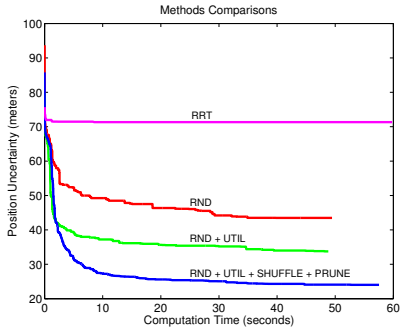


Fig. 8. Comparison of several tree search algorithms (averaged over 100 Monte Carlo runs). A standard RRT algorithm is not suitable for optimal planning since it quickly converges to and remains at a low quality solution. The algorithm RND converges asymptotically but the rate decreases with computation 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)

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 state space of the vehicle is $\mathcal{M}=SO(3)\times\mathbb{R}^3\times\mathbb{R}^6$ with $\mu=((R,p),(\omega,v))$ and the four control inputs are $u=(u_c,u_\psi,\gamma_p,\gamma_r)$.

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} R\widehat{\omega} \\ Rv \end{bmatrix}, \tag{31}$$

$$\begin{bmatrix} \mathbb{J}\dot{\omega} \\ m\dot{v} \end{bmatrix} = \begin{bmatrix} \mathbb{J}\omega\times\omega \\ mv\times\omega+R^T(0,0,-9.81m) \end{bmatrix} + f(u), \tag{32}$$

where the map $\widehat{\cdot}:\mathbb{R}^3\to\mathfrak{so}(3)$ and the control force $f(u)$ are defined as

$$\widehat{\omega}=\begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix}, \quad f(u)=\begin{bmatrix} d_t\sin\gamma_r u_c \\ d_t\sin\gamma_p\cos\gamma_r u_c \\ d_r u_\psi \\ \sin\gamma_p\cos\gamma_r u_c \\ -\sin\gamma_r u_c - u_\psi \\ \cos\gamma_p\cos\gamma_r u_c \end{bmatrix}.$$

The local motion planning method corresponding to **Connect** is based on sequencing of precomputed motion primitives which satisfy the dynamics (31)–(32). This is accomplished using a maneuver automaton [32] using a set of primitives which 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 minimum number of primitives can then be computed instantly in closed form through inverse kinematics. The terrain is represented using a digital elevation map loaded from a file. Collision checking and avoidance is performed using the

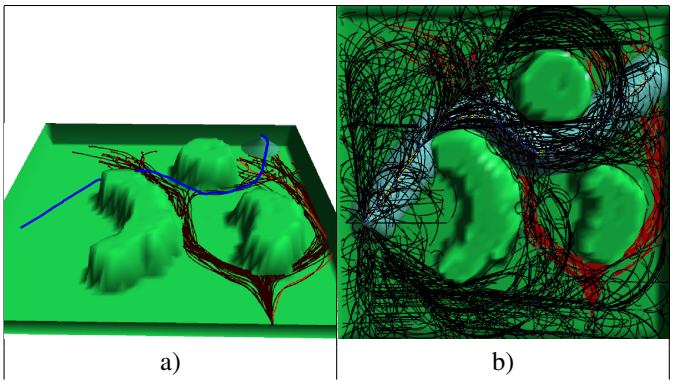


Fig. 9. The algorithm applied to the helicopter example described in §VIII-B showing a) the optimal helicopter trajectory; b) the constructed roadmap and sensor footprint along a path; c) close-up view along an edge of the roadmap.

Proximity Query Package (PQP) [33] 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 §II extended to 3-D. The helicopter is not permitted to fly above obstacles. Fig. 9 shows the resulting helicopter trajectory and a view of the constructed roadmap.

IX. Conclusion and Future Directions

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 [34, 35] 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 [36]) that optimally accounts for the sampled data during optimization.

In addition, there could be interesting connections to the RRT* methodology [37] which also employs tree modification in the context of deterministic planning. While at the time of this work we did not consider links to RRT*, it would

be fruitful in the future to explore the analysis introduced in [37] to study and improve the performance of our proposed method. It would also be useful to explore connections to recent sampling-based methods for efficiently solving partially observable Markov decision processes (POMDPs) [38]. In particular, the significant speed-up achieved by adaptively approximating the belief search space [39] and factorization exploiting mixed observability [40] have similar flavor to the methods proposed by this work. An interesting direction is to combine such POMDP techniques with the proposed random tree methods in order to efficiently handle systems with complex dynamics and constraints.

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.

REFERENCES

[1] M. Kobilarov, J. E. Marsden, and G. S. Sukhatme, "Global estimation in constrained environments," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 24–41, 2011.

[2] H. Khalil, *Nonlinear System, 3rd edition*. Prentice Hall, 1996.

[3] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. D. Schutter, "A comparison of decision making criteria and optimization methods for active robotic sensing," in *Lecture Notes in Computer Science*, vol. LNCS 2542, 2003, pp. 316–324.

[4] J. de Geeter, J. de Schutter, H. Bruyninckx, H. van Brussel, and M. Decreton, "Tolerance-weighted L-optimal experiment design for active sensing," in *Proc. Conf. IEEE/RSJ Int Intelligent Robots and Systems*, vol. 3, 1998, pp. 1670–1675.

[5] O. Tremois and J.-P. Le Cadre, "Optimal observer trajectory in bearings-only tracking for maneuvering sources," *IEE Proceedings -Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 31–39, 1999.

[6] S. S. Singh, N. Kantas, B.-N. Vo, A. Doucet, and R. J. Evans, "Simulation-based optimal sensor scheduling with application to observer trajectory planning," *Automatica*, vol. 43, p. 817 830, 2007.

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

[8] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-theoretic coordinated control of multiple sensor platforms," in *Proc. IEEE International Conference on Robotics and Automation ICRA '03*, vol. 1, 14–19 Sept. 2003, pp. 1521–1526.

[9] S. Paris and J.-P. Le Cadre, "Planning for terrain-aided navigation," in *Proc. Fifth Int Information Fusion Conf*, vol. 2, 2002, pp. 1007–1014.

[10] R. He, S. Prentice, and N. Roy, "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)*, Los Angeles, CA, 2008, pp. 1814–1820.

[11] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Robotics: Science and Systems*, 2005.

[12] R. Sim and N. Roy, "Global A-optimal robot exploration in slam," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2005*, 2005, pp. 661–666.

[13] L. Mihaylova, J. D. Schutter, and H. Bruyninckx, "A multisine approach for trajectory optimization based on information gain," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 231–243, 2003.

[14] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski, *Nonsmooth Analysis and Control Theory*. Springer, 1998.

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

[16] R. Y. Rubenstein and D. P. Kroese, *Simulation and the Monte Carlo Method*. Wiley, 2008.

[17] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics, 2007.

[18] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Springer, 2001.

[19] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, pp. 1–32, March 1996. [Online]. Available: http://dx.doi.org/10.1162/evco.1996.4.1.1

[20] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 18–28, 1997.

[21] R. Vaidyanathan, C. Hocaoglu, T. S. Prince, and R. D. Quinn, "Evolutionary path planning for autonomous air vehicles using multi-resolution path representation," in *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, vol. 1, 2001, pp. 69–76.

[22] C. Hocaoglu and A. C. Sanderson, "Planning multiple paths with evolutionary speciation," *IEEE Transations on Evolutionary Computing*, vol. 5, no. 3, pp. 169–191, 2001.

[23] G. Erinc and S. Carpin, "A genetic algorithm for nonholonomic motion planning," in *Proc. IEEE Int Robotics and Automation Conf*, 2007, pp. 1843–1849.

[24] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Press, 1991.

[25] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.

[26] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo methods in practice*, 2001.

[27] P. Del Moral, *Feynman-Kac Formulae. Genealogical and interacting particle systems with applications.* Springer-Verlag, 2004.

[28] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*.

MIT Press, June 2005.

[29] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics: Science and Systems*, 2005.

[30] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized algorithms for analysis and control of uncertain systems*. Springer, 2004.

[31] M. Vidyasagar, "Randomized algorithms for robust controller synthesis using statistical learning theory," *Automatica*, vol. 37, no. 10, pp. 1515–1528, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1016/S0005-1098(01)00122-4

[32] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, dec 2005.

[33] S. Gottschalk, M. C. Lin, and D. Manocha, "OBB-Tree: A hierarchical structure for rapid interference detection," *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, vol. 30, pp. 171–180, 1996.

[34] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization*. Springer, 2004.

[35] F. Celeste, F. Dambreville, and J.-P. L. Cadre, "Optimal path planning using cross-entropy method," in *2006 9th International Conference on Information Fusion*, 2007, pp. 1 – 8.

[36] M. Kobilarov, "Cross-entropy randomized motion planning," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[37] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[38] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, 1998. [Online]. Available: http://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf

[39] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proc. Robotics: Science and Systems*, 2008.

[40] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "POMDPs for robotic tasks with mixed observability," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.