Cross-Entropy Randomized Motion Planning

Marin Kobilarov

***Abstract*—This paper is concerned with motion planning for nonlinear robotic systems operating in constrained environments. Motivated by recent developments in sampling-based motion planning and Monte Carlo optimization we propose a general randomized path planning method based on sampling in the space of trajectories. The idea is to construct a probability distribution over the set of feasible paths and to perform the search for an optimal trajectory through importance sampling. At the core of the approach lies the cross-entropy method for estimation of rare-event probabilities. The algorithm recursively approximates the optimal sampling distribution which guides the set of sampled trajectories towards regions of progressively lower cost until converging to a delta distribution at the optimum. Our main goal is to provide a framework for consistent adaptive sampling correlating the spatial structure of trajectories and their computed costs. The approach is illustrated with two simple examples–a point mass vehicle and the Dubins car, and is then applied to a simulated helicopter flying optimally in a 3-D terrain.**

I. Introduction

Consider a robotic vehicle with state $x \in \mathcal{X}$ controlled using actuator inputs $u \in \mathcal{U}$, where $\mathcal{X}$ is the state space and $\mathcal{U}$ denotes the set of controls. The vehicle dynamics is described by the function $f : \mathcal{X} \times \mathcal{U} \to T\mathcal{X}$ defined by

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

which is used to evolve the vehicle state forward in time. In addition, the vehicle is subject to constraints arising from actuator bounds and obstacles in the environment. These constraints are expressed through the vector of inequalities

$$F(x(t)) \geq 0, \tag{2}$$

for all $t \in [0, t_f]$, where $t_f > 0$ is the final time of the trajectory. The goal is to compute the optimal controls u^* and time t_f^* driving the system from its initial state $x_0 \in \mathcal{X}$ to a given goal region $\mathcal{X}_f \subset \mathcal{X}$, i.e.

$$\begin{aligned} (u^*, t_f^*) &= \operatorname*{argmin}_{u, t_f} \int_0^{t_f} C(u(t), x(t)) \mathrm{dt}, \\ \text{subject to } & \dot{x}(t) = f(x(t), u(t)), \\ & F(x(t)) \geq 0,\ x(0) = x_0,\ x(t_f) \in \mathcal{X}_f \end{aligned} \tag{3}$$

for all $t \in [0, t_f]$ and where $C : \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is a given cost function. A typical cost function includes a time component and a control effort component, i.e. $C(u, x) = 1 + \lambda \|u\|^2$ where $\lambda \geq 0$ is a chosen weight.

The problem (3) has no closed-form solution since both the dynamics (1) and constraints (2) are nonlinear. Gradient-based optimization is not suitable unless a good starting guess is chosen since the constraints (2) impose many local minima. In addition, constraints corresponding to arbitrary obstacles can be non-smooth and require special differentiation [3] to guarantee convergence. An alternative is to discretize the vehicle state space $\mathcal{X}$, e.g. using a grid and generate candidate paths by transitioning between adjacent cells. Such an approach is generally limited to few dimensions, e.g. $\dim(\mathcal{X}) \leq 3$ and to systems with very simple dynamics, e.g. an unconstrained point mass in the plane. This is due to the exponential (both in state dimension and planning horizon T) size of the search space, also known as the *curse of dimensionality*.

Since the motion planning problem (3) is computationally NP-complete in general [13] one has to resort to approximation algorithms. *Sampling-based motion planning* has become an established methodology in this context. The basic idea is to construct a graph structure with nodes corresponding to states and with edges satisfying the dynamics (1) and constraints (2). In essence, the graph is regarded as a finite approximation of the infinite set of feasible trajectories. The optimal control problem is then solved approximately through graph search. The two main families of such methods are *rapidly-exploring dense trees* (RDT) [13] and *probabilistic roadmaps* (PRM) [2]. In recent years considerable attention has been given to augmenting the basic sampling-based motion planners with strategies for accelerating the search. A common realization is that the complex problem can be solved effectively through a proper balance of *exploration and exploitation* [17, 18] and by guiding the algorithm based on acquired information.

Motivated by these developments we propose a technique for adaptive sampling which uses the cost of computed solutions to guide the search. In contrast to standard motion planning, we consider *sampling from the space of trajectories* rather than sampling nodes from the state space $\mathcal{X}$ alone. In essence, the problem is solved as a random search by optimally exploiting the gathered information from prior samples. At the core of our approach lies the *cross-entropy* (CE) method [20, 9] initially developed for estimation of rare-event probabilities and later employed as a general optimization framework. The CE method is widely applicable and is used to successfully solve complex combinatorial problems such as the minimum graph cut or the traveling salesman problems. The basic idea behind applying the CE approach to motion planning is to recursively iterate the two steps:

1) Generate trajectories sampled from a distribution and compute their costs
2) Update the distribution using a subset of "high-quality" trajectories

until the set of sampled trajectories becomes concentrated around the optimum, or equivalently until the distribution has converged to a delta function. The scheme is general and converges to an optimum assuming that enough feasible trajectories can be sampled. As with most randomized methods there is no guarantee that this would be the global optimum but

with high likelihood the global optimum can be approached if enough samples are generated.

We first reformulate the motion planning problem into a finite dimensional constrained optimization in §II. In §III we provide a quick introduction to the required probabilistic techniques for the CE approach. The main cross-entropy algorithm is then given in §IV. Two simple examples are used to illustrate the theory in §V. The link to standard sampling-based motion planning is then elaborated in §VI. The method is finally applied to a complex high-dimensional problem: a simulated helicopter navigating optimally in an urban terrain.

II. Problem Formulation

A trajectory recording the controls and states over the time interval $[0, t_f]$ is denoted by the function $\pi : [0, t_f] \to \mathcal{U} \times \mathcal{X}$, i.e. $\pi(t) = (u(t), x(t))$ for all $t \in [0, t_f]$. A given control curve $u : [0, t_f] \to \mathcal{U}$ determines a unique state trajectory $x : [0, t_f] \to \mathcal{X}$ by evolving the dynamics from an initial state $x_0 \in \mathcal{X}$. This is more formally defined through the *integral flow* of the ODE (1) denoted by $\Phi_u : \mathcal{X} \times \mathbb{R} \to \mathcal{X}$ in terms of

$$x(t) = \Phi_u(x_0, t), \forall t > 0. \tag{4}$$

The space of all trajectories originating at point x_0 and satisfying the dynamics is denoted

$$\mathcal{P} = \{\pi : t \in [0, +\infty] \to (u(t), x(t)) \mid x(t) = \Phi_u(x_0, t)\}.$$

Consider a finite-dimensional parametrization of trajectories in terms of vectors $z \in \mathcal{Z}$ where $\mathcal{Z} \subset \mathbb{R}^{n_z}$ is the *parameter space*. Assume that the parametrization is given by the function $\varphi : \mathcal{Z} \to \mathcal{P}$ according to

$$\pi = \varphi(z) \equiv \varphi_z.$$

The (control, state) tuples along a trajectory parametrized by z are written as $\pi(t) = \varphi_z(t)$. Note that we do not impose any restrictions on $\mathcal{Z}$ other that $\varphi(\mathcal{Z}) \subset \mathcal{P}$. Ideally $\varphi(\mathcal{Z})$ should be a "good" approximation of $\mathcal{P}$, i.e. for any given $\pi \in \mathcal{P}$ there should always exist a $z \in \mathcal{Z}$ for which $\varphi(z)$ is "close" to π.

Let $\kappa : \mathcal{U} \times \mathcal{X} \to \mathcal{X}$ project onto the state component, i.e. $\kappa(u, x) = x$. The *constrained parameter space* $\mathcal{Z}_{\text{con}} \subset \mathcal{Z}$ is the set of parameters satisfying the boundary conditions and constraints and is defined by

$$\mathcal{Z}_{\text{con}} = \{z \in \mathcal{Z} \mid F(\kappa(\varphi_z(t))) \geq 0, \kappa(\varphi_z(t_f)) \in \mathcal{X}_f.\}, \tag{5}$$

for some $t_f > 0$ and all $t \in [0, t_f]$. Define the *cost function* $J : \mathcal{Z} \to \mathbb{R}$ according to

$$J(z) = \int_0^{t_f} C(\varphi_z(t))dt.$$

Problem (3) can now be equivalently restated as finding the optimal t_f^* and $(x^*, u^*) = \varphi(z^*)$ such that

$$z^* = \arg\min_{z \in \mathcal{Z}_{\text{con}}} J(z). \tag{6}$$

We consider problems over a continuous set $\mathcal{Z}$ since the discrete case is too constraining for problems involving dynamics. In addition, for systems with very complicated dynamics $\mathcal{Z}$ can be a product of discrete and continuous spaces as in the hybrid system abstraction considered in §VII-B.

III. Background on Monte Carlo Optimization

We provide a quick introduction to the probabilistic framework for solving the optimization (6) through sampling trajectories from $\mathcal{Z}_{\text{con}}$. Our development follows closely [19] using notation adapted to our setting.

A. Importance Sampling

Consider the estimation of the following expression

$$\ell = \mathbb{E}_p[H(Z)] = \int H(z)p(z)dz, \tag{7}$$

where $H : \mathcal{Z} \to \mathbb{R}$ is some non-negative performance metric and p is the probability density of Z. Assume that there is another dominating [1] probability density q which is easy to evaluate and sample from, such as a Gaussian. The integral (7) can be expressed as

$$\ell = \int H(z)\frac{p(z)}{q(z)}q(z)dz = \mathbb{E}_q\left[H(z)\frac{p(z)}{q(z)}\right]. \tag{8}$$

The density q is called the *importance density* and can be used to evaluate the integral using i.i.d. random samples $Z_1, ..., Z_N$ from q so that

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} H(Z_i)\frac{p(Z_i)}{q(Z_i)} \tag{9}$$

is an unbiased estimator of ℓ. An important question is then how to select a good density q. The most natural choice is the density that minimizes the variance of the estimator $\hat{\ell}$, i.e.

$$q^* = \arg\min_q \mathbb{V}_q\left(H(Z)\frac{p(Z)}{q(Z)}\right),$$

the solution to which is

$$q^*(z) = \frac{H(z)p(z)}{\ell} \tag{10}$$

since $\mathbb{V}_{q^*}(\ell) = 0$. The density q^* is called the *optimal importance sampling density*. Of course, this density is only hypothetical and cannot be implemented in practice since it involves the value of ℓ which is what is being estimated in the first place!

A natural way to find a density q that is closest to q^* is in the Kullback-Leibler (KL) sense, i.e. with minimum cross-entropy distance between q^* and q. The KL distance between any two given distributions q and p is defined by

$$\mathfrak{D}(p, q) = \int p(z)\ln p(z)dz - \int p(z)\ln q(z)dz. \tag{11}$$

and the required q solves the following optimization

$$\min_q \mathfrak{D}(q^*, q). \tag{12}$$

We next consider the case when Z has a pdf $p(\cdot; \bar{v})$ belonging to some parametric family $\{p(\cdot; v), v \in \mathcal{V}\}$ where $\bar{v}$ is the true or nominal parameter. For instance, this could be a mixture of Gaussians. It is natural to consider an importance density q

[1] the density q dominates Hp when $q(z) = 0 \Rightarrow H(z)p(z) = 0$.

from the same family. Its optimal parameter v is found through the parametric optimization

$$\min_v \mathfrak{D}(q^*, p(\cdot, v))$$

This is equivalent to maximizing with respect to v

$$\int H(z)p(z,\bar{v})\ln p(z,v)dz,$$

which is obtained using (10) and (11). In other words, the optimal importance density parameter v^* can be found as

$$v^* = \operatorname*{argmax}_v \mathbb{E}_{\bar{v}}[H(Z)\ln p(Z,v)]. \tag{13}$$

Finally, the optimal parameter can be approximated numerically by

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N}\sum_{i=1}^{N} H(Z_i)\ln p(Z_i,v), \tag{14}$$

where $Z_1, ..., Z_n$ are i.i.d. samples from $p(\cdot, \bar{v})$.

B. Estimation of Rare-Event Probabilities

Consider the estimation of the probability that a trajectory $z \in \mathcal{Z}_{con}$ sampled from $p(\cdot;\bar{v})$ has a cost $J(z)$ smaller than a given constant γ. It is defined as

$$\ell = \mathbb{P}_{\bar{v}}(J(Z)\le\gamma) = \mathbb{E}_{\bar{v}}[I_{\{J(Z)\le\gamma\}}], \tag{15}$$

where $I_{\{\cdot\}}$ is the indicator function. This can be computed approximately using (9) according to

$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I_{\{J(Z_i)\le\gamma\}}\frac{p(Z_i;\bar{v})}{p(Z_i;v)},$$

where $Z_1, ..., Z_N$ are i.i.d. samples from $p(\cdot, v)$. In order to determine the optimal v for this computation we can employ (14) to obtain

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N}\sum_{i=1}^{N} I_{\{J(Z_i)\le\gamma\}}\ln p(Z_i,v), \tag{16}$$

where $Z_1, ..., Z_N$ are i.i.d. samples from $p(\cdot, \bar{v})$. The problem is that when $\{J(Z)\le\gamma\}$ is a rare event, this approximation is meaningless because there will be almost no samples z with $J(z)\le\gamma$ and $\hat{\ell}$ will be incorrectly estimated as zero!

The idea behind the CE method is to employ a multilevel approach using a sequence of parameters $\{v_j\}_{j\ge0}$ and levels $\{\gamma_j\}_{j\ge1}$. At the end the sequence converges to the optimal v^* which then can be used to estimate the integral $\hat{\ell}$ correctly. The procedure starts by drawing N samples $Z_1, ..., Z_N$ using an initial parameter v_0, for instance $v_0 = \bar{v}$. Let ϱ be a small number, e.g. $10^{-2} \le \varrho \le 10^{-1}$. The value γ_1 is set to the ϱ–th quantile of $H(Z)$, i.e. γ_1 is the largest real number for which

$$\mathbb{P}_{v_0}(H(Z)\le\gamma_1) = \varrho.$$

The level γ_1 can be computed approximately by sorting the costs of the samples $J(Z_1), ..., J(Z_N)$ in an increasing order, say $J_1 \le ... \le J_N$, and setting $\hat{\gamma}_1 = J_{\lceil\varrho N\rceil}$. The optimal parameter v_1 for level $\hat{\gamma}_1$ is then estimated using (14) by replacing γ with $\hat{\gamma}_1$.

Note that the samples with costs $J_1, ..., J_{\lceil\varrho N\rceil}$ will also be the samples used to estimate v_1. They form the *elite set*, i.e. the ϱ-fraction of the N samples with the best costs. The procedure then iterates to compute the next γ_i and v_i and terminates when $\gamma_i \le \gamma$. At this point we set $v = v_i$ as the optimal parameter corresponding to the originally given level γ and the probability of $J(Z)\le\gamma$ is computed using v. In summary, each iteration of the algorithm perform two steps, starting with v_0,

1) *Sampling and updating of* γ_j: Sample $Z_1, ..., Z_n$ from $p(\cdot, \hat{v}_{i-1})$ and compute the ϱ-th quantile $\hat{\gamma}_t$.
2) *Adaptive Updating of* v_j: Compute $\hat{v}_j$ such that

$$\hat{v}_j = \operatorname*{argmin}_{v\in\mathcal{V}} \frac{1}{|\mathcal{E}_j|}\sum_{Z_k\in\mathcal{E}_j} \ln p(Z_k;v), \tag{17}$$

where $\mathcal{E}_j$ is the *elite* set of samples, i.e. samples Z_k for which $J(Z_k)\le\hat{\gamma}_j$.

C. Cross-Entropy Optimization

The idea behind the cross-entropy method is to treat the optimization (6) as an *estimation problem of rare-event probabilities*. Define the cost function optimum γ^* by

$$\gamma^* = \min_{z\in\mathcal{Z}_{\text{con}}} J(z).$$

Finding the optimal trajectory then amounts to iterating the rare-event simulation steps defined in §III-B until the cost γ_j approaches γ^*. Typically, after a finite number of iterations $p(\cdot, v_j)$ will approach a delta distribution and all samples Z_i will become almost identical. This signifies that the optimum has been found and z^* is set to the sample with lowest cost. Note that the term "optimal" should be used with caution because although the method explores the state space globally it might still converge to a local solution if, for instance, no samples were obtained near the true global value.

IV. Motion Planning

We now construct a general motion planning algorithm based on the cross-entropy method. We choose to represent the importance density using a Gaussian mixture model since it is a compact way to encode a rich set of trajectories across multiple homotopy classes.

The parameter space is $\mathcal{V} = (\mathbb{R}^{n_z}\times\mathbb{R}^{(n_z^2+n_z)/2})^K\times\mathbb{R}^K$ with elements $v = (\mu_1, \Sigma_1, ..., \mu_K, \Sigma_K, w_1, ..., w_K)$ corresponding to K mixture components with means μ_k, covariance matrices Σ_k (excluding identical elements due to the matrix symmetry) and weights w_k. The density is defined as

$$p(z;v) = \sum_{k=1}^{K}\frac{w_k}{\sqrt{(2\pi)^{n_z}|\Sigma_k|}}e^{-\frac{1}{2}(z-\mu_k)^T\Sigma_k^{-1}(z-\mu_k)},$$

where $\sum_{k=1}^{K} w_k = 1$. The number of mixture components K is chosen adaptively (see e.g. [4]). Even the simplest case with $K = 1$ is capable of solving complex multi-extremal problems globally. The complete algorithm is summarized below.

Algorithm 4.1: CE Motion Planning

1) Choose initial trajectory samples $Z_1, ..., Z_N$ so that the set $\hat{\mathcal{P}} = \{\varphi_{Z_i}\}_{i=1}^N$ approximates (sparsely) the set of feasible trajectories over $\mathcal{X}$; Set $j = 0$ and $\hat{\gamma}_0 = \infty$
2) Update $\hat{v}_j$ using (17) (e.g. by EM) over the elite set $\mathcal{E}_j = \{Z_i \mid J(Z_i) \leq \hat{\gamma}_j\}$
3) Generate samples $Z_1, ..., Z_N$ from $p(\cdot, v_j)|_{\mathcal{Z}_{\text{con}}}$ and compute the ϱ–th quantile $\hat{\gamma}_{j+1} = J_{\lceil \varrho N \rceil}$
4) If a stopping criteria is met then finish, otherwise set $j = j + 1$ and goto step (2)

There are several important points determining the success of the approach.

a) Initialization.: The initial set of samples should be generated (step (1)) to achieve a good coverage of $\mathcal{X}$. The samples do not need to have good quality in terms of the cost J, they only need to achieve a reasonable exploration of $\mathcal{X}$. They can be generated using an initial parameter v_0, i.e. $Z_i \sim p(\cdot, v_0)$ where v_0 can encode some a priori knowledge about the problem. A more general approach that does not rely on any problem assumptions is to perform uniform sampling over $\mathcal{Z}_{\text{con}}$. This corresponds to a high-entropy uninformative random start also proven effective in the context of mixture models [15]. An alternative is also to generate the initial set using a classical motion planning roadmap [13, 2].

b) Parameter Update.: The optimal parameter update (step (2)) is accomplished using an Expectation-Minimization (EM) algorithm [15] for $K \geq 2$. In the case $K = 1$ the components of $\hat{v}_j$ are updated simply as

$$\mu = \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} Z_k, \quad \Sigma = \frac{1}{|\mathcal{E}_j|} \sum_{Z_k \in \mathcal{E}_j} (Z_k - \mu)(Z_k - \mu)^T.$$

As the algorithm iterates, the uncertainty volume (i.e. the determinants of the covariances Σ_k) is shrinking towards a delta distribution. This might happen prematurely before reaching a good quality solution if, for instance, too few samples were used. To prevent such degeneracy it is useful to inject a small amount of noise with variances $\nu \in \mathbb{R}^{n_z}$ (see [1]) i.e. by setting $\Sigma_k = \Sigma_k + \text{diag}(\nu)$ for each $k = 1, ..., K$.

c) Sampling.: The sampling step (3) in Algorithm (4.1) is accomplished using a standard accept-reject argument and is given below only for convenience.

Generating Samples over $\mathcal{Z}_{con}$
1) Compute $A_k = \sqrt{\Sigma}_k$ for all $k = 1, ..., K$ and set $i = 1$
2) Choose $k \in \{1, ..., K\}$ proportional to w_k
3) Sample $r \sim \mathcal{N}_{n_z}(0, 1)$ and set $Z_i = \mu_k + A_k r$
4) if $Z_i \notin \mathcal{Z}_{con}$ then go to line (3)
5) if $i = N$ then finish; else set $i = i + 1$ and goto line (2)

Note that the only expensive operation is the square-root (using e.g. Cholesky decomposition) on line (1) but it is performed only K times *before* drawing all N samples. It is also possible to handle constrained sampling more efficiently through a local search to detect the boundary of $\mathcal{Z}_{\text{con}}$ and escape from $\mathcal{Z} \backslash \mathcal{Z}_{\text{con}}$.

V. Examples

Consider a physical workspace (i.e. an environment) denoted by $\mathcal{W}$, where $\mathcal{W} = \mathbb{R}^2$, or $\mathcal{W} = \mathbb{R}^3$ [12]. 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. Assume that the vehicle at state $x \in \mathcal{X}$ is occupying a region $\mathcal{A}(x) \subset \mathcal{W}$ and that the function $\textbf{prox}(\mathcal{A}_1, \mathcal{A}_2)$ 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 (2) is then to avoid obstacles, generally expressed as

$$F_1(x(t)) = \min_i \textbf{prox}(\mathcal{A}(x(t)), \mathcal{O}_i), \text{ for all } t \in [0, \infty]. \quad (18)$$

We demonstrate the cross-entropy method to two simple types of vehicles navigating in a planar obstacle field–a point mass double integrator and a Dubins car.

A. Piece-wise Cubic Spline

Consider a point mass vehicle operating in the plane with state space $\mathcal{X} = \mathbb{R}^2 \times \mathbb{R}^2$ and state $x = (q, v)$. It has a double-integrator dynamics given by $\dot{q} = v$ and $\dot{v} = u$ where the controls $u \in \mathcal{U} = \mathbb{R}^2$ are simply the accelerations. Assume that the cost function is of the form

$$C(u, x) = \|v\| + \lambda \|u\|^2,$$

i.e. a combination of the length of the path and its acceleration where $\lambda > 0$ controls the curve smoothness. The goal set consists of a single point $x_g \in \mathcal{X}$, i.e. $\mathcal{X}_g = \{x_g\}$.

1) Trajectory Generation: A trajectory is parametrized using m "knots" $(q_i, v_i) \in \mathbb{R}^4$, $i = 1, ..., m$. The parameter space is $\mathcal{Z} = \mathbb{R}^{4m}$ with state $z = (q_1, v_1, ..., q_m, v_m)$. Denote the given start state by $x_0 = (q_0, v_0)$ and denote the goal state by $x_{m+1} \equiv x_g$. In essence, the trajectory will be represented by $m + 1$ cubic splines joining the pairs of states $x_i \to x_{i+1}$, for $i = 0, ..., m$. Since the cost function does not involve time we can use simply set $t_f = 1$ and generate all trajectories over the time interval $[0, 1]$. Let $t_i = i/(m+1)$ denote the starting time of the segment beginning at state x_i. Over the interval $[t_i, t_{i+1}]$ for each $i = 0, ..., m$ the function $\varphi_z = (u, q, v)$ takes the form

$$\begin{aligned} u(t) &= 6a_i\bar{t} + 2b_i, \\ v(t) &= 3a_i\bar{t}^2 + 2b_i\bar{t} + c_i, \\ q(t) &= a_i\bar{t}^3 + b_i\bar{t}^2 + c_i\bar{t} + d_i, \end{aligned}$$

where $\bar{t} := t - t_i$, $c_i = 3q_{i+1} - 3q_i - 2v_i - v_{i+1}$, $d_i = -2q_{i+1} + 2q_i + v_i + v_{i+1}$.

2) Numerical Results: Consider an environment depicted on Fig. 1. It contains one concave obstacle and several homotopy classes which create an interesting planning scenario. Algorithm 4.1 is run using $N = 100$ and $\varrho = .1$ and one mixture component $K = 1$. Fig. 1 shows that it quickly converges to the globally optimal solution (known in advance) in a few iterations. Empirically Fig. 2 shows that the cost function decreases nearly exponentially in the number of iterations.

B. Dubins Car

The Dubins car is one of the simplest vehicle models with non-holonomic constraints. Its state space is $\mathcal{X} = SE(2)$ with state $x = (r_x, r_y, \theta)$ denoting the car position (r_x, r_y) and orientation θ. The dynamics is defined as

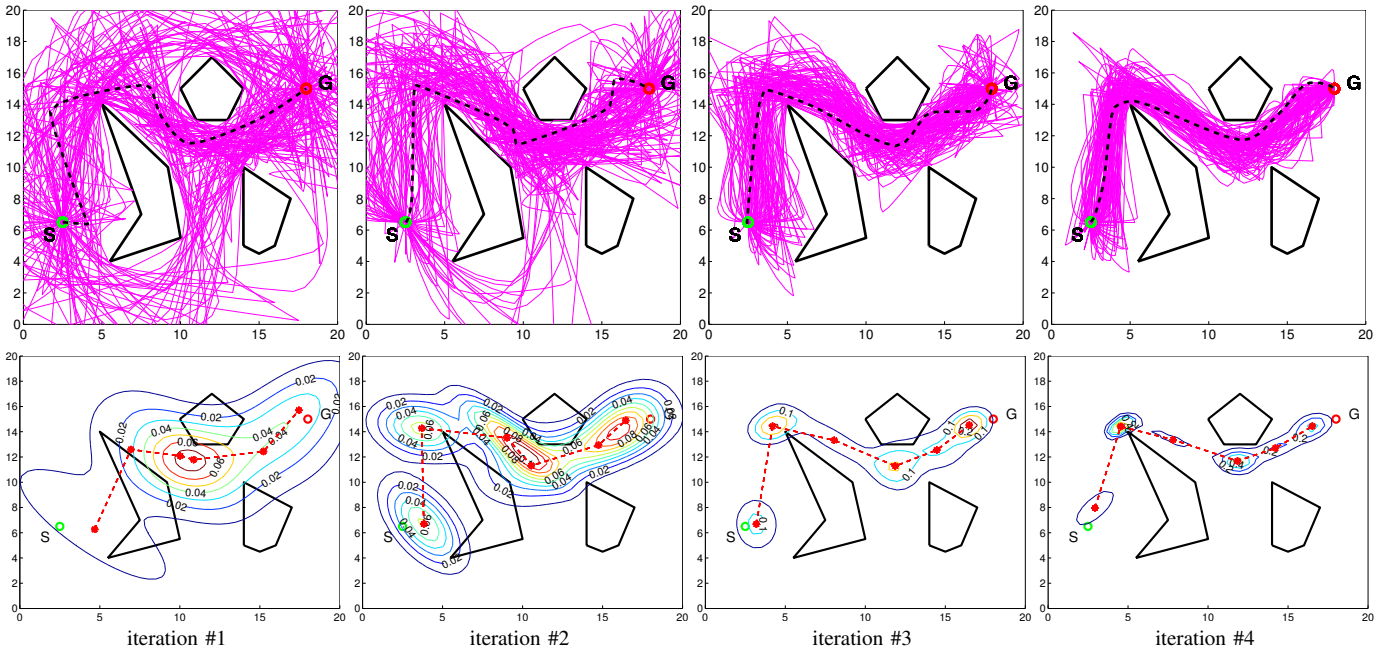


Fig. 1. The first four iterations of the CE algorithm 4.1, i.e. $j = 1, ..., 4$, applied to a point mass vehicle among obstacles. The upper plots show the sampled trajectories $\varphi(Z_1), ..., \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). The lower plots visualize $p(\cdot, \hat{v}_j)$ as the level sets of another density over the (x, y)-position space. The density of each point corresponds to the density of the best trajectory passing through it. The algorithm starts from a nearly uniform state space exploration and converges towards a delta distribution at the global optimum.

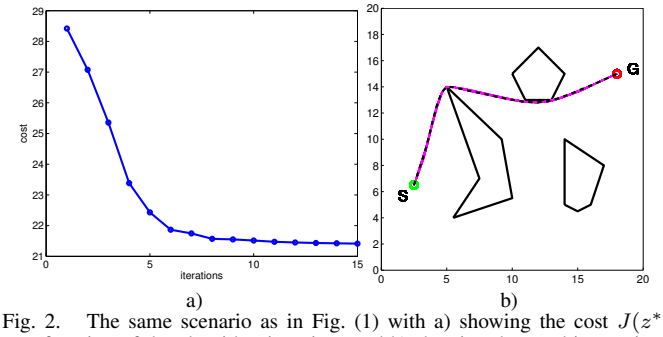


Fig. 2. The same scenario as in Fig. (1) with a) showing the cost $J(z^*)$ as a function of the algorithm iterations and b) showing the resulting optimal path $\varphi(z^*)$.

$$\dot{r}_x = v\cos\theta, \quad \dot{r}_y = v\sin\theta, \quad \dot{\theta} = u, \tag{19}$$

where $v > 0$ is a given constant speed and u is the controllable turn rate. The control space is $\mathcal{U} = [-\tan\phi_{max}, \tan\phi_{max}]$, where $\phi_{max} \in (0, \pi/2)$ is the maximum steering angle. The Dubins car fails the small-time local controllability (STLC) test which creates interesting non-trivial control problems 13.

1) Trajectory Generation: A trajectory satisfying (19) can be represented using a set of connected *primitives* which are either straight lines with velocity v or arcs with radius v/u in position space. A primitive is represented by its constant control u and time duration, denoted by $\tau > 0$. Assume that a trajectory is parametrized using m such primitives, i.e.

$$z = (u_1, \tau_1, ..., u_m, \tau_m) \in \mathbb{R}^{2m}.$$

The ending time of the execution of the i-th primitive, denoted t_i, is computed as $t_i = \sum_{k=1}^{i} \tau_k$ for $i = 1, ..., m$ with $t_0 = 0$. During time interval $t \in [t_i, t_{i+1}]$ the parametrization $\varphi_z = (u, r_x, r_y, \theta)$ takes the form

$$u(t) = u_{i+1}$$

$$r_x(t) = \begin{cases} r_x(t_i) + \frac{v}{u_{i+1}}(\sin(\theta_i + \Delta t_i u_{i+1}) - \sin\theta_i), & \text{if } u_{i+1} \neq 0 \\ r_x(t_i) + v\Delta t_i \cos\theta_i, & \text{otherwise} \end{cases}$$

$$r_y(t) = \begin{cases} r_y(t_i) + \frac{v}{u_{i+1}}(\cos\theta_i - \cos(\theta_i + \Delta t_i u_{i+1})), & \text{if } u_{i+1} \neq 0 \\ r_y(t_i) + v\Delta t_i \cos\theta_i, & \text{otherwise} \end{cases}$$

$$\theta(t) = \theta_i + (\Delta t_i) u_{i+1},$$

where $\Delta t_i = t - t_i$ and $\theta_i = \theta(t_i)$.

2) Numerical Results: Consider the same environment as in Fig. 1 and the same algorithm parameters as in §V-A. A trajectory is represented using $m = 6$ primitives. Fig. 3 shows the algorithm progression and the cost evolution and final solution are shown on Fig. 4.

Note that the number of primitives m can be adaptively chosen. For instance, if the condition $\alpha_i = \alpha_{i+1}$ persists during iteration then we simply remove α_{i+1}, set $m = m - 1$ and merge the times $\tau_i = \tau_i + \tau_{i-1}$. A primitive α_i can also be removed if the condition $\tau_i \approx 0$ persists. In that respect the algorithm has the capacity to determine an *optimal number* of different primitives.

VI. Links to Sampling-based Motion Planning

It is interesting to note that the CE approach effectively constructs a sequence of parametric models of sets of progressively "better" trajectories, i.e. with decreasing costs. Since a trajectory is a set of states this model can also be regarded as a density over the state space $\mathcal{X}$. A state $x \in \mathcal{X}$ has a high importance density, say $p_{\mathcal{X}}(x)$, if it belongs to a trajectory

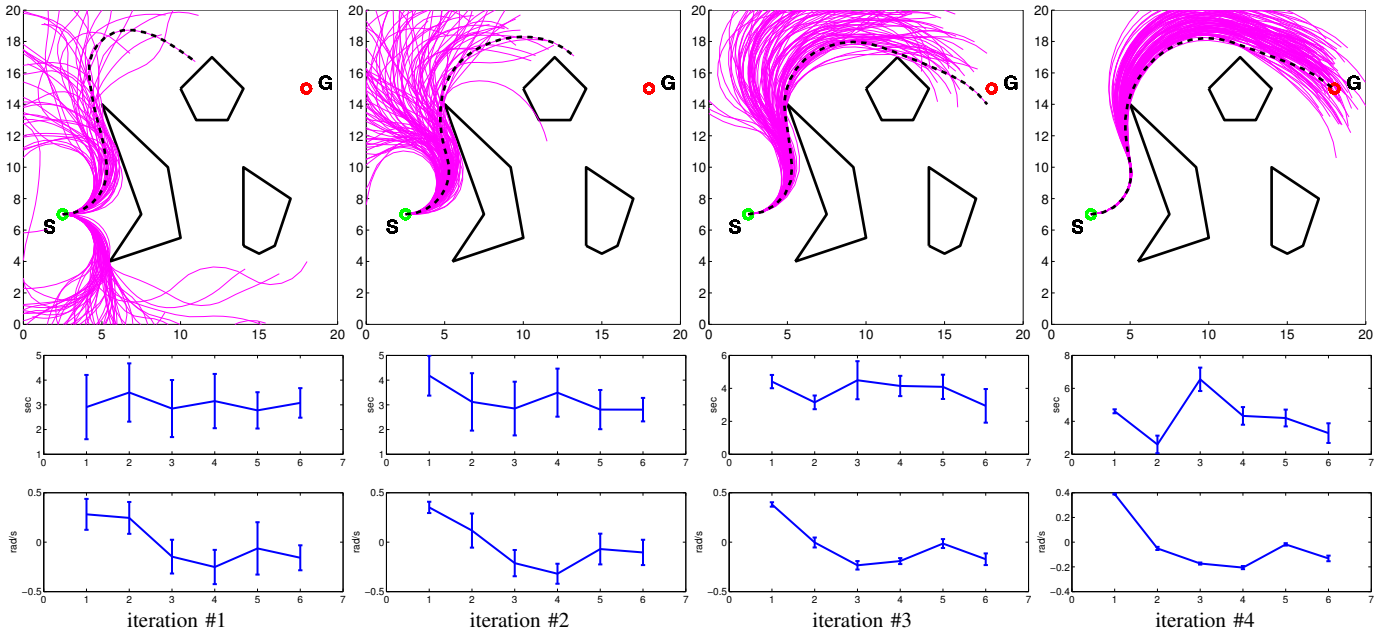


Fig. 3. The first four iterations of the CE algorithm 4.1 applied to the Dubins car. The upper plots show the sampled trajectories $\varphi(Z_1), ..., \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). The lower plots visualize $p(\cdot, \hat{v}_j)$ by plotting the components of the mean μ and standard deviations extracted from the diagonal of Σ. A total of $m = 6$ primitives were used and the lower plots shows the graphs of the time durations (i, τ_i) and commanded velocities (i, ω_i).

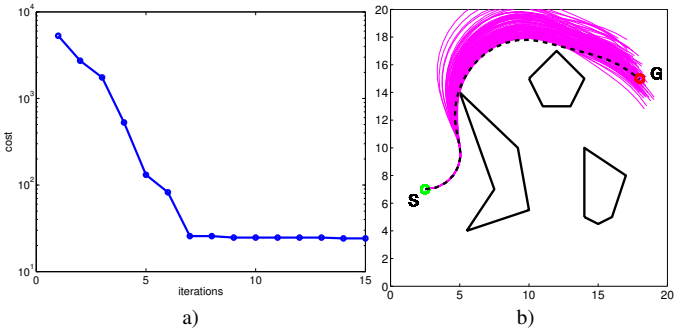


Fig. 4. The resulting cost $J(z^*)$ of the Dubins car path optimization (see Fig. (3)) is shown on a) while the resulting optimal path $\varphi(z^*)$ on b).

$\varphi_z \in \mathcal{P}$ with high density $p(z)$. Such a sampling density is defined as

$$p_{\mathcal{X}}(X; v) = \eta \cdot \max_{Z \in \mathcal{Z}_{\text{con}}} \{p(Z; v) \mid X \in \kappa(\varphi_Z)\}, \qquad (20)$$

where X is a random variable over $\mathcal{X}$ and $\eta > 0$ is a normalizing constant. For instance, the lower plots of Fig. 1 show the evolution of $p_{\mathcal{X}}$ restricted to the position space. The issue of how to properly *sample and connect nodes* lies at the core of almost all classical motion-planning methods. While various successful methods for improved sampling have been proposed (e.g. [10, 11, 7, 8, 18, 14]) the *optimal* nodes selection remains a hard and unsolved problem. In that respect, the density $p_{\mathcal{X}}$ is in fact an approximation to the optimal density for sampling nodes. A fruitful future direction is therefore to employ (20) for adaptive sampling in the context of probabilistic roadmap methods.

VII. Application to Systems with Symmetries

The cross-entropy method can naturally be applied to systems with symmetries [5]. Symmetries and invariance are important properties found in most robotic locomotion systems [16]. The associated structure of the dynamics can be exploited to produce simple motions termed primitives which can be regarded as building blocks for constructing complicated motions to achieve any given state. We first provide a quick review of primitives, then introduce the cross-entropy motion planning method, and finally develop an application to a simulated helicopter with non-trivial dynamics operating in a mountaineous terrain.

A. Planning Using Primitives

Planning using primitives relies on the notion of *trajectory invariance*. A primitive is a trajectory that can be transformed, e.g. rotated or translated in space, while remaining a feasible motion respecting the dynamics. Invariance is defined through the action of a Lie group G acting on the state space $\mathcal{X}$. Action by an element $g \in G$ on $x \in \mathcal{X}$ is denoted by $g \cdot x$. An invariant trajectory is such that the flow of the dynamics of the system is G-invariant, i.e.

$$g \cdot \Phi(x_0, t) = \Phi(g \cdot x_0, t).$$

Two trajectories $\pi_1, \pi_2 \in \mathcal{P}$ are *equivalent* if they differ by a group action and a time translation, i.e. if there exists a $g \in G$ and a time t' such that

$$(x_1(t), u_1(t)) = (g \cdot x_2(t - t'), u_2(t - t')),$$

for all t along the trajectory.

An important property of primitives is that they can be concatenated in order to create longer, more complicated

motions. This happens only if they are compatibile. Two trajectories π_1 and π_2 are *compatible*, denoted $\pi_1 C \pi_2$ if there exists some $g \in G$ s.t.

$$x_1(t_{1,f}) = g \cdot x_2(0),$$

where $t_{1,f} > 0$ is the final time of π_1. Any invariant trajectory can be regarded as a primitive. There are two main classes of primitives that we consider: trim primitives and maneuvers.

1) Trim Primitives: Trim trajectories correspond to continuously parametrized steady-state motions. Formally, a trajectory $\alpha : t \in [0, t_f] \to (x_\alpha(t), u_a(t))$ is a *trim primitive* if

$$x_\alpha(t) = \exp(t\xi_\alpha) \cdot x_\alpha(0), u_\alpha(t) = u_\alpha, \forall t \in [0, t_f],$$

i.e. if the motion is a finite flow along a left invariant vector field with constant control inputs. The vector $\xi_\alpha \in \mathfrak{g}$ is the body-fixed representation of the invariant vector field of trim α where $\mathfrak{g} \sim T_eG$ denotes the Lie algebra of the group G. The set of all trim primitives is denoted by $\mathcal{T}$.

A one-parameter family of trim motions corresponding to α can be constructed according to

$$\alpha(\tau) : t \in [0, \tau] \to (\exp(t\xi_\alpha) \cdot x_\alpha(0), u_\alpha) \tag{21}$$

where τ denotes the *coasting time*, and the set of all such primitives is defined by $\mathcal{T}_\alpha = \{\alpha(\tau), \tau \geq 0\}$. The displacement of a trim primitive α with coasting time τ is simply $g_\alpha = \exp(\tau\xi_\alpha)$.

2) Maneuvers: Maneuvers are designed to efficiently switch from one steady-state motion to another. Therefore they are defined to be compatible form left and right with trim primitives. The set of maneuvers is denoted $\mathcal{M}$. Formally, a maneuver π satisfies

$$\pi \in \mathcal{M} \Leftrightarrow \exists \alpha, \beta \in \mathcal{T} : \alpha C \pi C \beta$$

The displacement as a result of executing the maneuver is denoted g_π.

3) Primitives Library: A library of primitives constitutes a *maneuver automaton*, i.e. a finite-state machine with states corresponding to trim conditions and transitions corresponding to maneuvers. In practice, a library is constructed by first selecting a discrete set of trim vector fields denoted $\Xi \subset \mathfrak{g}$, e.g. by gridding a bounded subspace of interest in $\mathfrak{g}$. The steady state trim conditions $(x_\alpha(0), u_\alpha)$ are then computed for each $\xi_\alpha \in \Xi$. Finally, one or more maneuvers are designed to transition between any two trims α_1, α_2. The maneuvers can be computed by recording commanded motions of an actual or simulated system. In this work we design maneuvers through optimal control of nonlinear system models.

4) Trajectory Generation: Consider the task of generating a trajectory from a given state $x_0 \in \mathcal{X}$. It is typical to assume that this is a steady state corresponding to some trim primitive. Denoted the primitive by α_0 with steady state $x_{\alpha_0}(0)$. Then we have $x_0 = g_0 \cdot x_{\alpha_0}(0)$ for some $g_0 \in G$. Consider a sequence of trim primitives $\alpha_0, \alpha_1, ..., \alpha_N$ with coasting times $\tau_0, \tau_1, ..., \tau_N$ and connecting maneuvers $\pi_0, ..., \pi_{N-1}$. It forms the trajectory $\rho \in \mathcal{P}$ starting from x_0 defined by

$$\rho = \alpha_0(\tau_0)\pi_0\alpha_1(\tau_1)\pi_1 ... \alpha_N(\tau_{N-1})\pi_{N-1}\alpha_N(\tau_N). \tag{22}$$

The state along ρ are expressed, for $k = 1, ..., N$, by

$$\rho(t) = \begin{cases} (g_k \exp((t - t_k)\xi_{\alpha_k}) \cdot x_{\alpha_k}(0), u_{\alpha_k}), & t \in [t_k, t_k + \tau_k], \\ (g_k \exp(\tau_k \xi_{\alpha_k}) \cdot x_{\pi_k}(t'), u_{\pi_k}(t')), & t \in [t_k + \tau_k, t_{k+1}] \end{cases}, \tag{23}$$

where $g_k = g_0 \prod_{i=0}^{k-1} \exp(\tau_i \xi_{\alpha_i}) g_{\pi_i}$, $t_k = \sum_{i=0}^{k-1} (\tau_i + |\pi_i|)$, and $t' = t - t_k - \tau_k$.

In addition, the total group displacement along ρ is

$$g_\rho = g_N \exp(\tau_N \xi_{\alpha_N}). \tag{24}$$

5) Computing motion plans: Consider the task of finding a sequence of primitives driving the system from its initial state x_0 to a given final state $x_f \in \mathcal{X}$. Normally, these initial and final conditions are given in terms of steady states corresponding to trim primitives. Let α_0 and α_f denote the given boundary trims with steady states $x_{\alpha_0}(0)$ and $x_{\alpha_f}(0)$. Then we have $x_0 = g_0 \cdot x_{\alpha_0}(0)$ and $x_f = g_f \cdot x_{\alpha_f}(0)$ for some group elements $g_0, g_f \in G$. Computing a motion from x_0 to x_f then amounts to finding a proper sequence of primitives $\alpha_1, ..., \alpha_N$, coasting times $\tau_0, \tau_1, ..., \tau_N, \tau_f$, and connecting maneuvers $\pi_0, ..., \pi_N$. The sequence will form the trajectory $\rho \in \mathcal{P}$ defined by

$$\rho = \alpha_0(\tau_0)\pi_0\alpha_1(\tau_1)\pi_1 ... \alpha_N(\tau_N)\pi_N\alpha_f(\tau_f). \tag{25}$$

The total group displacement along ρ is

$$g_\rho = \left[\prod_{i=0}^{N} \exp(\tau_i \xi_{\alpha_i}) g_{\pi_i}\right] \exp(\tau_f \xi_{\alpha_f}) \tag{26}$$

and computing a motion from x_0 to x_f amounts to finding a motion plan ρ such that

$$g_\rho = g_0^{-1} g_f.$$

Solving this equation is equivalent to the problem of *kinematic inversion* in robotics. Therefore, the original differential control system was transformed [5] into a purely *kinematic system*, with the transformation being exact without any approximations.

6) Exact Trajectory Generation: A trajectory between two given boundary conditions x_0 and x_f can be uniquely parametrized using m trim primitives $\alpha_1, ..., \alpha_m$, the coasting times $\tau_0, ..., \tau_f$, and the required maneuvers to transition between trims. For simplicity, assume that there is exactly one maneuver between any two given trim primitives, i.e. there exists a unique $\pi_{\alpha\beta} \in \mathcal{M}$ such that $\alpha C \pi_{\alpha\beta} C \beta$ for any given $\alpha, \beta \in \mathcal{T}$. A trajectory is then parametrized only through its steady state properties. We choose to construct trajectories by sequencing m primitives from the start state x_0 and m primitives backwards from the final state x_f. The gap between the two evolved states is "patched" using a closed-form solution of n primitives where $n = \dim(G)$. The parameter space is $\mathcal{Z} = \mathfrak{g}^{2m+2} \times \mathbb{R}^{2m+2} \sim \mathbb{R}^{(n+1)(2m+2)}$ with elements

$$z = (\tau_0, \rho_{1:m}, \xi_{m+1}, \xi_{m+n-1}, \rho_{m+n:2m+n-1}, \tau_f), \tag{27}$$

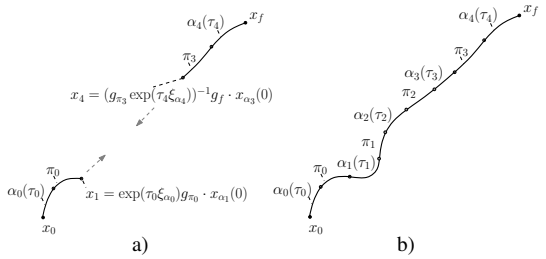


Fig. 5. Generating a trajectory between x_0 and x_f using primitives. The first step is to evolve a trajectory forward from the start and in reverse from the end (shown in a). A closed-form sequence of n trim primitives connected with maneuvers is then computed to match the evolved boundary conditions (shown in b).

where $\rho_{1:m} = (\xi_1, \tau_1, ..., \xi_m, \tau_m)$. The reason for the form of (27) is technical and related to matching trim conditions at the boundaries. Figure 5 illustrates the construction in the simplest case $m = 0$ which can be extended for any $m > 0$ is a straightforward manner. In particular, the sequence $\tau_0, \rho_{1:m}$, is evolved as described in (VII-A4) to reach state g'_0. The sequence $\rho_{m+n:2m+n-1}, \tau_f$ is evolved in the same way but in reverse to reach states g'_f. The remaining trim conditions ξ_{m+1}, ξ_{m+n-1} are then used as boundary conditions corresponding to g_o and g'_f to close the gap as described in §VII-A5.

B. Aerial Vehicle Application

Consider a small autonomous helicopter depicted in Fig. 6 operating in a 3-D terrain. The vehicle is modeled as a single underactuated rigid body with position $\mathbf{r} \in \mathbb{R}^3$ and orientation matrix $R \in \mathrm{SO}(3)$. Its *body-fixed* angular and linear velocities are denoted by $\omega \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$, respectively. The vehicle has mass m and principal moments of rotational inertia J_1, J_2, J_3 forming the inertia tensor $\mathbb{J} = \mathrm{diag}(J_1, J_2, J_3)$.

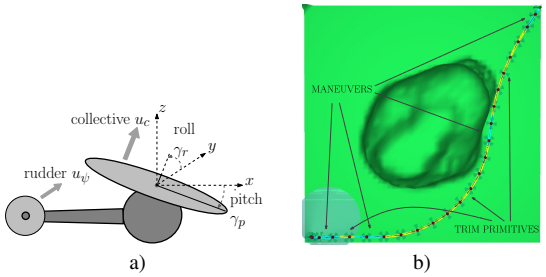


Fig. 6. Simplified helicopter model used in our tests (a). An example of a computed sequence of four maneuvers and three trim primitives is shown (b), connecting two zero-velocity states in the corners of the workspace and avoiding an obstacle in the center.

The 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 $\mathcal{X} = SE(3) \times \mathbb{R}^6 \times \mathbb{R}^2$ with $x = ((R, \mathbf{r}), (\omega, \mathbf{v}), \gamma)$.

The equations of motion are

$$\begin{bmatrix} \dot{R} \\ \dot{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} R\,\widehat{\omega} \\ R\,\mathbf{v} \end{bmatrix}, \tag{28}$$

$$\begin{bmatrix} \mathbb{J}\,\dot{\omega} \\ m\dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbb{J}\,\omega \times \omega \\ m\mathbf{v} \times \omega + R^T(0, 0, -9.81m) \end{bmatrix} + F(\gamma)u, \tag{29}$$

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

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

Since gravity is the only configuration-dependent term in the dynamics and is invariant to translations and rotations the z-axis, i.e. to transformations by the symmetry group $G = SE(2) \times \mathbb{R}$. Therefore, the helicopter motion generation is abstracted through the primitives framework developed for systems with symmetries in [5].

1) Trim Primitive: Trim motions therefore correspond to helices, circles, or straight lines in position space. Such curves are traced using velocity $\xi \in \mathfrak{se}(2)$ with components $(\omega'_z, v'_x, v'_y, v'_z)$ which are related to the body-fixed velocities v and ω through

$$\omega = R^T_{(\phi,\theta,0)}\omega', \qquad \mathbf{v} = R^T_{(\phi,\theta,0)}\mathbf{v}', \tag{30}$$

where $R_{(\phi,\theta,\psi)}$ denotes rigid body rotation parametrized by roll ϕ, pitch θ, and yaw ψ and where $\mathbf{v}' = (v'_x, v'_y, v'_z)$ and $\omega' = (0, 0, \omega'_z)$.

A group transformation $g \in G$ is described by components $(\Delta\psi, \Delta\mathbf{r})$ corresponding to yaw change $\Delta\psi$ and position change $\Delta\mathbf{r}$. The action $g \cdot x$ then takes the form

$$\begin{aligned} g \cdot x &= (\Delta\psi, \Delta\mathbf{r}) \cdot (\omega, \mathbf{v}, \phi, \theta, \psi, \mathbf{r}, \gamma) \\ &= (\omega, \mathbf{v}, \phi, \theta, \psi + \Delta\psi, R_{(0,0,\Delta\psi)}\mathbf{r} + \Delta\mathbf{r}, \gamma) \end{aligned}$$

An invariant trajectory has a constant body-fixed velocity. Therefore, in order to determine the invariance conditions for desired ω' and $\mathbf{v}'$ one sets $\dot{\mathbf{v}} = 0$ and $\dot{\omega} = 0$ in (29) and substitutes (30) into (29). In order to simplify the derivation of the trim conditions assume two identical inertia components $J_2 = J_3$ and no vertical velocity, i.e. $v'_z = 0$. A closed-form solution can be computed as

$$\begin{aligned} &\theta = 0, \quad u_y = 0, \quad \gamma_p = 0, \quad \gamma_r = 0, \\ &\phi = \arctan(-w'_z v'_x / g), \\ &u_c = m(g\cos\phi \; - w'_z v'_x \sin\phi). \end{aligned} \tag{31}$$

Finally, the exponential map of a given element $\xi = (\omega'_z, v'_x, v'_y, v'_z)$ is $\exp(t\xi) = (t\omega_z, \Delta r_x, \Delta r_x, tv_z)$ where

$$\Delta r_x = \begin{cases} (v'_x \sin t\omega'_z - v'_y(1 - \cos t\omega'_z))/\omega'_z, & \text{if } \omega'_z \neq 0 \\ tv'_x, \text{ otherwise,} \end{cases}$$

$$\Delta r_y = \begin{cases} (v'_x(1 - \cos t\omega'_z) + v'_y \sin t\omega'_z)/\omega'_z, & \text{if } \omega'_z \neq 0 \\ tv'_y, \text{ otherwise.} \end{cases}$$

While we consider only planar trims $v'_z = 0$ the connecting maneuvers evolve in 3-D position space. Trims are stored by discretizing the space of allowable velocities (ω', v'_x, v'_y).

2) Maneuvers.: Maneuvers are computed to connect two trimmed states. Let the map $\varpi : X \to X\backslash G$ subtract out the invariant coordinates from a given state, i.e. $\varpi((\omega, \mathbf{v}, \phi, \theta, \psi, \mathbf{r}, \gamma)) = (\omega, \mathbf{v}, \phi, \theta, 0, \mathbf{0}, \gamma)$. A maneuver between two given primitives with states x_α and x_β is computed through nonlinear optimization of a trajectory whose start and end satisfy the given trim conditions. This is defined through the following procedure:

$$\begin{aligned} &\textit{Compute}: && T;\ x : [0,T] \to \mathcal{X};\ u : [0,T] \to \mathcal{U} \\ &\textit{minimizing}: && J(x,u,T) = \int_0^T (1 + \lambda \|u(t)\|^2)\, \mathrm{d}t, \\ &\textit{subject to}: && \varpi(x(0)) = x_\alpha, \varpi(x(T)) = x_\beta, \\ & && \text{dynamics eq. (28),(29) for all } t \in [0,T]. \end{aligned}$$

for some $\lambda > 0$ which adds regularity to the problem. The optimizations were performed offline and all trims and maneuvers are organized and saved in a library which is loaded at run-time providing instant look-up during planning.

3) Motion Planning: The cross-entropy algorithm 4.1 is implemented using by evolving m primitives forward from the start state x_0 and m primitives backward in reverse from the goal state x_f. A total of 5 primitives are used for closing the gap between the evolved states (since $\dim(\mathrm{SE}(2)) = 3$ and two maneuvers are required between the three trims). A trajectory consisting of the minimum number of five primitives is computed instantly in closed form through inverse kinematics. Fig. 6 shows an example of such a sequence of primitives. 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) [6] that compute closest distance between arbitrary polyhedra and was used to implement the function **prox** defined in (18). Trims are sampled from a continuous Gaussian mixture and the samples are projected onto the discrete grid of predefined trims. The exact details are omited due to lack of space.

4) Numerical Results: The algorithm is tested in a digital terrain corresponding to the training facility in Ft. Benning. The helicopter is not permitted to fly above buildings. Fig. 7 shows several iterations of the algorithm and the final optimized path between two given states. The first iteration took five seconds of computation time while the remaining iterations took less than a second. CPU time is dominated by attempting to sample trajectories which collide with obstacles. As the informedness of $p(Z; v)$ increases fewer infeasible trajectories are tried during sampling and each iteration completes in a few milliseconds.

VIII. Conclusion

The paper addresses the motion planning problem through a randomized optimization in the continuous space of feasible trajectories. Adaptive importance sampling using the cross-entropy method is employed to guide the search towards optimal solution regions. Empirical results show that the proposed motion planning algorithm can handle non-trivial problems as long as feasible trajectories can be sampled successfully. Future work will focus on deriving formal convergence rates. A promising direction in that respect is to establish a relationship between existing motion planning probabilistic completeness results [2] and existing bounds on the complexity of stochastic programming [20]. As a result it would be interesting to consider augmenting existing graph-based planners with the proposed cross-entropy adaptive sampling for optimally guiding the exploration.

References

[1] Zdravko Botev and Dirk P. Kroese. Global likelihood optimization via the cross-entropy method with an application to mixture models. In *Winter Simulation Conference*, pages 529–535, 2004.

[2] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005.

[3] Francis H. Clarke, Yuri S. Ledyaev, Ronald J. Stern, and Peter R. Wolenski. *Nonsmooth Analysis and Control Theory*. Springer, 1998.

[4] M. A. F. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002. doi: 10.1109/34.990138.

[5] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, dec 2005.

[6] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 30:171–180, 1996.

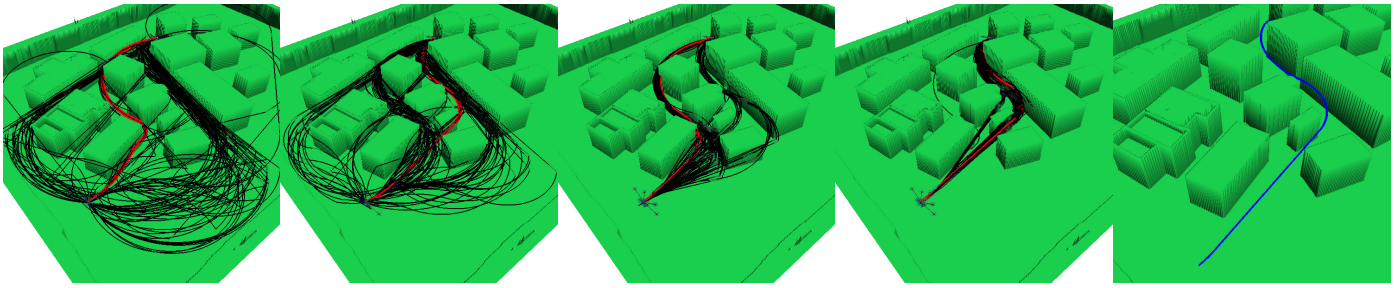
[7] D. Hsu, G. Sanchez-Ante, and Zheng Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2005*, pages 3874–3880, 2005. doi: 10.1109/ROBOT.2005.1570712.

[8] David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, 25(7):627–643, 2006.

[9] Dirk Kroese, Sergey Porotsky, and Reuven Rubinstein. The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, 8:383–407, 2006. ISSN 1387-5841.

[10] H. Kurniawati and D. Hsu. Workspace importance sampling for probabilistic roadmap planning. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2004)*, volume 2, pages 1618–1623, 2004.

[11] A. M. Ladd and L. E. Kavraki. *Fast Tree-Based Exploration of State Space for Robots with Dynamics*, pages 297–312. Springer, STAR 17, 2005.



iteration #1 iteration #2 iteration #5 iteration #8 optimal path

Fig. 7. Several iterations of the CE algorithm applied to the helicopter scenario. As the optimization proceeds the set of samples concentrates in the homotopy classes with minimum trajectory cost.

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

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

[14] Yanbo Li and K. E. Bekris. Balancing state-space coverage in planning with dynamics. In *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, pages 3246–3253, 2010. doi: 10.1109/ROBOT.2010.5509481.

[15] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. John Wiley & Sons, Inc., 2002.

[16] James Ostrowski. Computing reduced equations for robotic systems with constraints and symmetries. *IEEE Transactions on Robotics and Automation*, pages 111–123, 1999.

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

[18] M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008*, pages 2812–2817, 2008. doi: 10.1109/ROBOT.2008.4543636.

[19] Reuven Y. Rubenstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2008.

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