The International Journal of
Robotics Research
31(7) 855–871

DOI: 10.1177/0278364912444543
ijr.sagepub.com

Cross-entropy motion planning

Marin Kobilarov

Abstract

This paper is concerned with motion planning for non-linear robotic systems operating in constrained environments. A method for computing high-quality trajectories is proposed building upon recent developments in sampling-based motion planning and stochastic optimization. The idea is to equip sampling-based methods with a probabilistic model that serves as a sampling distribution and to incrementally update the model during planning using data collected by the algorithm. At the core of the approach lies the cross-entropy method for the estimation of rare-event probabilities. The cross-entropy method is combined with recent optimal motion planning methods such as the rapidly exploring random trees (RRT) in order to handle complex environments. The main goal is to provide a framework for consistent adaptive sampling that correlates the spatial structure of trajectories and their computed costs in order to improve the performance of existing planning methods.*

Keywords

motion planning, nonlinear control, importance sampling, cross-entropy, stochastic optimization, RRT

1. Introduction

Consider an agile robotic vehicle navigating in a natural environment. The vehicle motion is constrained due to its kinematics and dynamics and due to obstacles in the environment. The task is to compute an open-loop trajectory that reaches a desired goal region optimally under the assumption that perfect models of the robot and the environment are available. Motion planning is a key requirement for autonomous systems and solving this problem is of central importance in robotics.

In general, the problem cannot be solved in closed form since both the dynamics and constraints can be non-linear. Gradient-based optimization is not suitable unless a good starting guess is chosen since the constraints impose many local minima. In addition, constraints corresponding to arbitrary obstacles can be non-smooth and require special differentiation (Clarke et al. 1998) to guarantee convergence. An alternative is to discretize the vehicle state space, e.g. using a grid and generate candidate paths by transitioning between adjacent cells. Such an approach is computationally intractable if the state space has more than a few dimensions and is limited 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) size of the search space, also known as the 'curse of dimensionality'.

Since the motion planning problem is computationally NP-complete in general (LaValle 2006) 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 and constraints. 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) (LaValle 2006) and probabilistic roadmaps (PRM) (Choset et al. 2005). Such sampling-based methods are probabilistically complete, i.e. the probability of failing to find a solution, if it exists, approaches zero at least asymptotically in the number of iterations.

The property of sampling-based methods of interest in this work is optimality. Standard rapidly exploring random trees (RRT) methods can quickly explore the space but typically provide a solution far from optimal while standard PRM methods asymptotically approach the optimal

California Institute of Technology, 2543 Wellesley Avenue, Los Angeles, CA

Corresponding author:
Marin Kobilarov, California Institute of Technology, 2543 Wellesley Avenue, Los Angeles, CA 90064, USA.
Email: marin@cds.caltech.edu

trajectory but at an exponentially slow rate (due to the increased amount of vertices and edges) which in higher dimensions becomes computationally intractable. The optimal RRT (RRT*) and optimal rapidly exploring random graph (RRG*) were recently proposed (Karaman and Frazzoli 2011) to overcome some of these limitations while retaining probabilistic completeness. The basic idea is to maintain the standard RRT exploratory properties while rewiring the structure until it satisfies local dynamic programming conditions on the set of vertices which result in asymptotic optimality.

The performance of sampling-based methods, regarding optimality, can be further improved from a different viewpoint. In particular, typically much effort is wasted in sampling nodes from parts of the state space that are unlikely to improve the current solution. Instead, it is possible to sample from a probabilistic model that incrementally identifies promising regions of the state space using the costs of trajectories that have already reached the goal. The purpose of this paper is to construct a method that combines sampling-based methods with such an adaptive sampling approach in order to exploit the collected information about the optimality of trajectories.

Employing adaptive or biased sampling is not new in motion planning. Such ideas have lead to more efficient algorithms using guided sampling to reduce the chance of colliding edges and accelerate state space exploration in order to find solutions more efficiently (e.g. Burns and Brock 2005b; Hsu et al. 2006; Kalisiak and van de Panne 2007; Li and Bekris 2010; Knepper and Mason 2011). In addition, several methods have achieved significant improvements by exploiting workspace information (Kurniawati and Hsu 2004), learning from previous motion planning runs (Hsu et al. 2005), or exploiting rather than discarding colliding samples (Denny and Amato 2011). A related viewpoint is to adaptively balance between exploring the space and exploiting collected information about collisions (Ladd and Kavraki 2005; Rickert et al. 2008). The unifying idea behind most of these methods is the construction of a deterministic or a probabilistic model (Burns and Brock 2005a; Zucker et al. 2008) that can be adjusted before and during execution to improve planning performance.

At the same time, probabilistic models also serve as a basis for stochastic optimization (Spall 2003). Among the many flavors and applications of stochastic optimization we mention methods based on global models incorporating past data (e.g. Moore and Schneider (1995) and Atkeson et al. (1997) in the context of plant optimization) or local stochastic gradient-based algorithms (Powell 2007) which include several recently developed methods for robotic trajectory optimization among obstacles (Ratliff et al. 2009; Theodorou et al. 2010; Kalakrishnan et al. 2011).

The idea behind the methods that we propose is to employ stochastic optimization of probabilistic models in the context of sampling-based planning. For this purpose we employ the cross-entropy (CE) method (Rubinstein and Kroese 2004; Kroese et al. 2006) originally developed for estimation of rare-event probabilities and later employed as a general optimization framework. The CE method is a stochastic optimization technique that can be either local or global depending on the chosen model and prior. It is widely applicable and is used to 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 samples from a distribution and compute their costs;
2. update the distribution using a subset of 'high-quality' samples;

until the set of samples becomes concentrated around the optimum, or equivalently until the distribution has approached a delta function. The scheme is general and is expected to converge to an optimum assuming that enough feasible trajectories can be sampled. Yet, the exact number of samples required for approaching a global optimum is difficult to determine. Even though general theoretical convergence of the CE method has been shown (Homem-de Mello 2007; Margolin 2005; Costa et al. 2007; Hu et al. 2007; Hu and Hu 2009) actual rates of convergence, sample complexity, or precise performance guarantees remain open problems.

1.1. Contributions

This work builds upon recent developments in optimal sampling-based planning and stochastic optimization to develop a new method aimed at producing lower cost trajectories through optimally estimated adaptive sampling distribution. From the point of view of motion planning, one can consider the CE method as a way to optimize the distribution for sampling vertices. From the point of view of stochastic optimization, the motion planning component provides feasible samples (trajectories) that are otherwise prohibitively expensive to generate due to complex constraints. Thus, the proposed combined technique utilizes the strengths of both methods to compute trajectories that have lower costs after fewer iterations. The practical contribution of this paper is to spell out the details of two new algorithms combining the RRT* method (Karaman et al. 2011) with the CE method for robotic trajectory optimization (Kobilarov 2011). The difference between the two new methods is whether sampling occurs in the state space or in the space of parametrized trajectories. We provide empirical results of the performance of each approach and demonstrate marked improvement over existing methods. The major limitations of the approach lie in the increased computational time, the lack of a systematic procedure for choosing the best statistical model in view of the system and environment for a given problem, and the fact that the method is only useful in

scenarios in which multiple trajectories to the goal can be computed during the algorithm execution.

1.2. Organization

The motion planning problem is formulated in Section 2. An overview of the probabilistic techniques required for the CE method is given in Section 3.1 followed by a quick background on optimal sampling-based methods, in particular RRT*, in Section 3.2. The methods in Section 3.2 include a slight modification of the original version required by the proposed approach. The CE method applied to trajectory optimization is given in Section 4. The new CE motion planning algorithms are developed in Section 5 and illustrated with simple examples. A more detailed empirical analysis and comparisons using a double integrator in three dimensions and a simple aerial vehicle are given in Section 6.

2. Problem formulation

Consider a robotic vehicle with state trajectory $x : [0, T] \to \mathcal{X}$ controlled using actuator inputs $u : [0, T] \to \mathcal{U}$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ denotes the set of controls, and $T > 0$ is the final time of the trajectory. The state and controls at time $t > 0$ are denoted by $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$, respectively. The vehicle dynamics satisfies the ordinary differential equation (ODE)

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

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 $t \in [0, T]$. The goal is to compute the optimal controls u^* and time T^* driving the system from its initial state $x_0 \in \mathcal{X}$ to a given goal region $\mathcal{X}_g \subset \mathcal{X}$, i.e.

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

for all $t \in [0, T]$ 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. $C(u(t), x(t)) = 1 + \lambda \|u(t)\|^2$ where $\lambda \geq 0$ is a chosen weight.

3. Background

The proposed methodology is based on two main ingredients: the CE method and optimal sampling-based motion planning. Therefore, we first describe the general CE method and then a slightly modified version of RRT* to fit in our framework.

3.1. Cross-entropy method

The CE method can be regarded as an importance sampling solution to the problem of estimating rare events. Let Z denote a random variable defined over a space $\mathcal{Z}$. The rare event of interest in this work is finding a parameter z with a real-valued cost $J(z)$ which happens to be very close to the cost of an optimal parameter z^*. Therefore, as will be explained below, the rare-event estimation is equivalent to the global optimization of $J(z)$. Our development follows closely (Rubenstein and Kroese 2008).

3.1.1. Importance sampling Consider the estimation of the following expression

$$\ell = \mathbb{E}_p[H(Z)] = \int H(z)\, p(z)\, dz, \tag{4}$$

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 (4) can be expressed as

$$\ell = \int H(z) \frac{p(z)}{q(z)} q(z)\, dz = \mathbb{E}_q \left[H(z) \frac{p(z)}{q(z)} \right]. \tag{5}$$

The density q is called the 'importance density' and can be used to evaluate the integral using independent and identically distributed (i.i.d.) random samples $Z_1, \ldots, Z_N$ from q so that

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} H(Z_i) \frac{p(Z_i)}{q(Z_i)} \tag{6}$$

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

$$q^*(z) = \frac{H(z)\, p(z)}{\ell} \tag{7}$$

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 CE distance between q^* and q. The KL distance between any two given distributions q and p is defined by

$$\mathrm{KL}(p \,\|\, q) = \int p(z) \ln p(z)\, dz - \int p(z) \ln q(z)\, dz \tag{8}$$

and the required q solves the following optimization

$$\min_q \mathrm{KL}(\, q^* \parallel q)\,. \tag{9}$$

We next consider the case when Z has a probability density function (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 from the same family. Its optimal parameter v is found through the parametric optimization

$$\min_v \mathrm{KL}(\, q^* \parallel p(\,\cdot\,,v)\,)$$

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 (7) and (8). In other words, the optimal importance density parameter v^* can be found as

$$v^* = \operatorname*{argmax}_v \mathbb{E}_{\bar{v}}[H(Z) \ln p(Z,v)\,]. \tag{10}$$

Finally, the optimal parameter can be approximated numerically by

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N} \sum_{i=1}^{N} H(Z_i) \ln p(Z_i,v)\,, \tag{11}$$

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

3.1.2. Estimation of rare-event probabilities Consider the estimation of the probability that a parameter $z \in \mathcal{Z}$ sampled from $p(\,\cdot\,;\bar{v})$ has an associated cost $J(z)$ smaller than a given constant γ. It is defined as

$$\ell = \mathbb{P}_{\bar{v}}(J(Z) \le \gamma) = \mathbb{E}_{\bar{v}}[I_{\{J(Z)\le\gamma\}}], \tag{12}$$

where $I_{\{\cdot\}}$ is the indicator function. This can be computed approximately using (6) 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 $Z_1,\ldots,Z_N$ are i.i.d. samples from $p(\,\cdot\,,v)$. In order to determine the optimal v for this computation we can employ (11) to obtain

$$\hat{v}^* = \operatorname*{argmax}_{v\in\mathcal{V}} \frac{1}{N} \sum_{i=1}^{N} I_{\{J(Z_i)\le\gamma\}} \ln p(Z_i,v)\,, \tag{13}$$

where $Z_1,\ldots,Z_N$ 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 $Z_1,\ldots,Z_N$ using an initial parameter v_0, for instance $v_0=\bar{v}$. Let ϱ be a small number, e.g. $10^{-2}\le\varrho\le10^{-1}$. The value γ_1 is set to the ϱth 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 $J(Z_1),\ldots,J(Z_N)$ in an increasing order, say $J_1\le\cdots\le J_N$, and setting $\hat{\gamma}_1 = J_{\lceil\varrho N\rceil}$. The optimal parameter v_1 for level $\hat{\gamma}_1$ is then estimated using (11) by replacing γ with $\hat{\gamma}_1$.

Note that the samples with costs $J_1,\ldots,J_{\lceil\varrho N\rceil}$ 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 $Z_1,\ldots,Z_n$ from $p(\,\cdot\,,\hat{v}_{i-1})$ and compute the ϱth quantile $\hat{\gamma}_t$.
2. *Adaptive updating of* v_j: Compute $\hat{v}_j$ such that

$$\hat{v}_j = \operatorname*{argmin}_{v\in\mathcal{V}} \frac{1}{|\mathcal{E}_j|} \sum_{Z_k\in\mathcal{E}_j} \ln p(Z_k;v)\,, \tag{14}$$

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

3.1.3. CE optimization The idea behind the CE method is to treat the optimization of $J(z)$ as an estimation problem of rare-event probabilities. Define the cost function optimum γ^* by

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

Finding the optimal trajectory then amounts to iterating the rare-event simulation steps defined in Section 3.1.2 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.

3.2. Sampling-based motion planning

The methods developed in this paper are based on the RRT*/RRG* algorithms. The development will be restricted

to RRT but can be analogously applied in the RRG setting as well. The main point is to perform sampling which exploits all information collected about the costs of trajectories during the algorithm execution. Following Karaman et al. (2011) we construct a RRT* algorithm and augment it with a simple extension: a connection is attempted between all newly added nodes and the goal in order to generate as many trajectories reaching the goal region $\mathcal{X}_g$ as possible. The costs of these trajectories will comprise the data used for adaptive sampling.

Algorithm 1: $\mathcal{T} \leftarrow \texttt{RRT}^*(\eta_0, \mathcal{X}_g)$

```
1  T ← InitializeTree()
2  T ← InsertNode(∅, η0, T)
3  N_g ← ∅
4  for i = 1 : N do
5      η_rand ← Sample(i, N_g)
6      η_nearest ← Nearest(T, η_rand)
7      (x_new, u_new, T_new) ← Steer(η_nearest, η_rand)
8      if ObstacleFree(x_new) then
9          N_near ← Near(T, η_new, |V|)
10         η_min = ChooseParent(N_near, η_nearest, x_new)
11         T ← InsertNode(η_min, η_new, T)
12         T ← Rewire(T, N_near, η_min, η_new)
13         (x_g, u_g, T_g) ← Steer(η_new, X_g)
14         if ObstacleFree(x_g) and x_g(T_g) ∈ X_g then
15             T ← InsertNode(η_new, η_g, T)
16             UpdateCostToGo(η_new, Cost(x_g))
17             UpdateCostToCome(η_g, CostToCome
18             (η_new) + Cost(x_g))
19             N_g ← N_g ∪ {η_g}
20 return T
```

The RRT* algorithm. The RRT* algorithm maintains a tree $\mathcal{T}$ of nodes. Each node $\eta \in \mathcal{T}$ contains a state $x \in \mathcal{X}$ and pointers to its parent node and the set of children nodes through the functions $\texttt{Parent}(\eta)$ and $\texttt{Children}(\eta)$. The functions $\texttt{CostToCome}(\eta)$ and $\texttt{CostToGo}(\eta)$ maintain the cost from the start to the node and from the node to the goal set $\mathcal{X}_g$, and are set to ∞ initially. A new node η_{new} is inserted into the tree to become a child of an existing node η_{current} using a function $\texttt{InsertNode}(\eta_{\text{current}}, \eta_{\text{new}}, \mathcal{T})$ which creates an edge between the two nodes and also updates $\texttt{CostToCome}(\eta_{\text{new}})$. The function $\texttt{Steer}(\eta_1, \eta_2)$ generates an optimal trajectory $x : [0, T] \rightarrow \mathcal{X}$ that attempts to drive the system between the two given states x_1 and x_2 for time T. Complete details of the algorithm can be found in Karaman et al. (2011) and Karaman and Frazzoli (2011).

A simple extension. For the purposes of this work the key points are to generate trajectories that reach the goal and

Algorithm 2: $\eta_{\min} \leftarrow \texttt{ChooseParent}(\mathcal{N}_{\text{near}}, \eta_{\text{nearest}}, x_{\text{new}})$

```
1  η_min ← η_nearest
2  c_min ← CostToCome(η_nearest) + Cost(x_new)
3  for η_near ∈ N_near do
4      (x', u', T') ← Steer(η_near, η_new)
5      if ObstacleFree(x') and x'(T') = η_new then
6          c' = CostToCome(η_near) + Cost(x')
7          if c' < c_min then
8              η_min ← η_near
9              c_min ← c'
10 return η_min
```

Algorithm 3: $\mathcal{T} \leftarrow \texttt{Rewire}(\mathcal{T}, \mathcal{N}_{\text{near}}, \eta_{\min}, x_{\text{new}})$

```
1  for η_near ∈ N_near \ {η_min} do
2      (x', u', T') ← Steer(η_new, η_near)
3      if ObstacleFree(x') and x'(T') = η_near and
4        CostToCome(η_new) + Cost(x') <
         CostToCome(η_near) then
5          T ← Reconnect(η_new, η_near, T)
6          UpdateCostToCome(η_near, CostToCome
7          (η_new) + Cost(x'))
8  return T
```

to update the cost-to-come and cost-to-go parameters while the tree grows and rewires itself. This is accomplished by attempting to connect every newly added node not only to the existing tree but also to the goal region. Lines 13–18 in Algorithm 1 were added to accomplish this. The leaf nodes of all paths connecting to the goal are stored in a list $\mathcal{N}_g$ which will be used for adaptive sampling (see Section 5) and are passed as an argument to the function `Sample` on line 5. Two new procedures `UpdateCostToGo` (Algorithm 4) and `UpdateCostToCome` (Algorithm 5) are included which are called during rewiring and every time the goal is reached. The minimum cost of trajectories that reach the goal and pass through a given vertex η can then be computed by $\texttt{CostToCome}(\eta) + \texttt{CostToGo}(\eta)$. This value will be use used in the CE method for adaptive sampling in Section 5.

Algorithm 4: $\texttt{UpdateCostToGo}(\eta, c)$

```
1  while CostToGo(η) > c do
2      CostToGo(η) ← c
3      (η, x) ← Parent(η)
4      if η = ∅ then
           return
5      c = c + Cost(x)
```

Algorithm 5: `UpdateCostToCome`(η, c)

```
if CostToCome(η) > c then
    CostToCome(η) ← c
    N = Children(η)
    foreach η_child ∈ N do
        UpdateCostToCome(η_child, c + Cost((η, η_child)))
```

4. CE trajectory optimization

We next present the CE method for robotic trajectory optimization. In addition to describing the core approach (Kobilarov 2011) we focus on the key points of the algorithm required for successful implementation. In addition, we mention its shortcomings that motivate the development of the CE RRT methods.

The CE method is suitable for non-linear and high-dimensional systems operating in relatively uncluttered obstacle environments such as car-like or helicopter robots navigating among buildings or canyons. The method is based on sampling in parametrized trajectory space; while this has resulted in robust and efficient optimization in various settings, it also imposes important limitations. In particular, more complicated obstacles such as those associated with narrow passages significantly shrink the feasible regions in trajectory space and thus, in the absence of a good prior, can render the method intractable due to the large number of rejected samples. This is also one of the motivations for developing the main methods in this work (Section 5) that combining CE with optimal motion planning.

4.1. Trajectory parametrization

A trajectory recording the controls and states over the time interval $[0, T]$ is denoted by the function $\pi : [0, T] \to \mathcal{U} \times \mathcal{X}$, i.e. $\pi(t) = (u(t), x(t))$ for all $t \in [0, T]$. A given control curve $u : [0, T] \to \mathcal{U}$ determines a unique state trajectory $x : [0, T] \to \mathcal{X}$ by evolving the dynamics from an initial state $x_0 \in \mathcal{X}$ (under standard regularity conditions on the ODE (1)). Let $\tau(\pi)$ denote the duration of a given trajectory π. The space of all trajectories originating at point x_0 and satisfying the dynamics is denoted by

$$\mathcal{P} = \{\pi : t \in [0, T] \to (u(t), x(t)) \mid \dot{x}(t) = f(x(t), u(t)), \; x(0) = x_0, \; T > 0\}.$$

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

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

The (control, state) tuples along a trajectory parametrized by z are written as $\pi(t) = \varphi(z, t)$. In addition, it is useful to define the functions $\varphi_x(z, t)$ and $\varphi_u(z, t)$ which extract only the state $x(t)$ or the control $u(t)$, respectively, for given parameter z and time t. The time duration of a trajectory parametrized by z is given by $\tau(z)$. In addition, there is a function $\psi : \mathcal{P} \to \mathcal{Z}$ which extracts a set of parameters from a given trajectory, i.e. $z = \psi(\pi)$. Note that φ and ψ are not necessarily the inverse of each other since there can be many different parametrization choices yielding the same trajectory.

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

$$\mathcal{Z}_{\text{con}} = \{z \in \mathcal{Z} \mid F(\varphi_x(z, t)) \geq 0, \varphi_x(z, T) \in \mathcal{X}_g.\}, \quad (15)$$

for some $T > 0$ and all $t \in [0, T]$. Define the 'cost function' $J : \mathcal{Z} \to \mathbb{R}$ according to

$$J(z) = \int_0^T C(\varphi(z, t)) \, dt. \quad (16)$$

Problem (3) can now be solved approximately by finding T^* and $(x^*, u^*) = \varphi(z^*)$ such that

$$z^* = \arg\min_{z \in \mathcal{Z}_{\text{con}}} J(z). \quad (17)$$

This optimization will be solved through the CE optimization described in Section 3.1.

The correct choice of parametrization φ in general is non-trivial and depends on the problem. The basic requirement is to retain reachability and controllability properties of the original system. There are two general types of parametrizations: through primitives and through a finite number of states along the trajectory connected using a steering method.

Primitives. One general approach is to use $z = (u_1, \tau_1, \ldots, u_m, \tau_m)$ where each u_i, for $1 \leq i \leq m$, is a constant control input applied for duration τ_i. This is an example of a simple 'primitive', e.g. a motion with constant control u_i. Conditions for resolution completeness of planning with such primitives have been established (Yershov and LaValle 2011). Yet, many systems, for instance with inherently unstable dynamics, cannot be directly captured in this representation and require a more complicated type of primitives termed 'maneuvers' to achieve transitions between simpler primitives (Frazzoli et al. 2005). In addition, certain systems have special structural properties of the dynamics such as stable/unstable manifolds or limit cycles that could also be abstracted through special parameters in addition to control inputs.

States. Exact and near-optimal steering methods exist for many robotic locomotion systems, i.e. it is possible to compute a unique curve that satisfies given boundary conditions. This can be accomplished with the help of primitives, or by exploiting properties such as differential flatness

(see e.g. Murray et al. 1995; Murray and Sastry 1993). For such systems a trajectory can be parametrized directly as a sequence of states, e.g. $z = (x_1, \ldots, x_m)$ since the curves between x_0 and x_1, x_1 and x_2, etc. are computed using steering and thus the whole trajectory originating from x_0, passing through $x_1, \ldots, x_m$, and reaching $\mathcal{X}_g$ is uniquely determined by z.

Mixed parametrization. For certain systems steering can be solved through inverse kinematics of a properly chosen sequence of primitives. Thus, it is possible to employ either primitives or states and internally reconstruct one from the other. The helicopter example is one such system.

Examples of each of these three different representation are given in Section 4.5. In general, there is a complicated trade-off between the dimensionality of $\mathcal{Z}$, its resolution completeness, optimality gap due to quantization, the ability to handle complicated environments, and the resulting efficiency of the optimization. Determining provably good parametrization requires an in-depth study beyond the empirical evidence that we provide.

4.2. Choosing a probabilistic model

The CE method falls in the category of global optimization method based on probabilistic models (Zhigljavsky and Zilinskas 2008). The key point is that at every iteration it transforms the model to shift probability mass in low-cost regions. The underlying importance density we choose is a Gaussian mixture model (GMM) since it is a compact way to encode multiple trajectories across multiple homotopy classes. A GMM is chosen for computational convenience since the CE density estimation can be performed using well-established expectation–minimization (EM). On the other hand, a GMM is limited since it can only capture as many local regions in the space as the number of the components in its mixture. Yet, a favorable property is that each Gaussian component can be regarded as an approximation of a local second-order model of the objective function centered at each mean (i.e. by regarding the covariance as the inverse Hessian of the cost). This could explain its fast convergence in the vicinity of a local optimum observed in our examples (in the absence of obstacles).

In addition, the choice of Gaussian distributions is reinforced by the close links between the CE method and two other families of recent stochastic optimization methods, in particular[2] covariance matrix adaptation (CMA) evolution strategies (Igel et al. 2006) and estimation of distribution algorithms (EDA) (Larrañaga and Lozano 2002; Pelikan et al. 2002), as well as related ideas in earlier machine learning literature (e.g. De Bonet et al. 1996). While CE, CMA, and EDA can have different flavors depending on the chosen models and parameter values, their practical implementation is nearly identical in the single Gaussian case.

Algorithm 6: CE motion planning.

Initialization:

0.1 Compute the optimal trajectory reaching the goal by ignoring the constraints, i.e. $z^* = \min_{z \in \mathcal{Z}} J(z)$ such that $\varphi_x(z^*, \tau(z^*)) \in \mathcal{X}_g$

0.2 Set matrix Σ so that the region $\{\varphi_x(z,t) \mid (z - z^*)^T \Sigma (z - z^*) < 2,\ 0 \le t \le \tau(z)\} \subset \mathcal{X}$ covers the reachable state space of interest

0.3 Choose initial samples $Z_1, \ldots, Z_N$ from $\texttt{Normal}(z^*, \Sigma)$; set $j = 0$ and $\hat{\gamma}_0 = \infty$

Iteration:

1. Update $\hat{v}_j$ using (14) (e.g. by EM) over the elite set $\mathcal{E}_j = \{Z_i \mid J(Z_i) \le \hat{\gamma}_j\}$
2. Generate samples $Z_1, \ldots, Z_N$ from $p(\cdot, v_j)\,|_{\mathcal{Z}_{\text{con}}}$ and compute the ϱth quantile $\hat{\gamma}_{j+1} = J_{\lceil \varrho N \rceil}$
3. If $j > 0$ and $\text{KL}(p(\cdot, v_{j-1}) \,||\, p(\cdot, v_j)) < \epsilon$ then finish, otherwise set $j = j + 1$ and goto step (1)

4.3. A parametric density algorithm

A general trajectory optimization algorithm based on the CE method is next constructed. 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 $v = (\mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K, w_1, \ldots, w_K)$ 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

$$p(z; v) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^{n_z} |\Sigma_k|}} e^{-\frac{1}{2}(z-\mu_k)^T \Sigma_k^{-1} (z-\mu_k)}, \qquad (18)$$

where $\sum_{k=1}^{K} w_k = 1$. The number of mixture components K can be fixed or chosen adaptively (see e.g. Figueiredo and Jain 2002). In the absence of complex constraints even the simplest case with $K = 1$ is capable of solving complex multi-extremal problems.

The complete algorithm is summarized in Algorithm 6.

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

Initialization. The initial set of samples should be generated to achieve a good coverage of $\mathcal{X}$. We assume that no prior knowledge about the problem is available. Therefore, initial sampling is achieved by ignoring the obstacles and computing an optimal trajectory z^* (step 0.1). A normal distribution with covariance Σ chosen to cover the reachable space of interest (step 0.2) is centered at z^* to obtain the initial sample set (step 0.3). When condition (0.2) cannot be easily solved the covariance can be determined by trial and error until the environment is covered.

Parameter update. The optimal parameter update (step (1)) is accomplished using an EM algorithm (McLachlan and Peel 2002) 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 Botev and Kroese 2004), i.e. by setting $\Sigma_k = \Sigma_k + \text{diag}(\nu)$ for each $k = 1, \ldots, K$.

Sampling. The sampling step (2) in Algorithm (6) 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 $k = 1, \ldots, K$ and set $i = 1$
2. Choose $k \in \{1, \ldots, K\}$ 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 $Z_i \in \mathcal{Z}_{con}$ then go to line (3)
5. if $i = N$ then finish; otherwise 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 $\mathcal{Z} \backslash \mathcal{Z}_{\text{con}}$.

Termination. The algorithm ends after the change in distribution (measured by KL-divergence) between iterations has become less than a given small constant ϵ. Alternatively, it can be terminated if the GMM covariances have nearly shrunk to zero (measured by their determinant). Very irregular cost functions will prevent such convergence and require separately monitoring decrease in the current optimum in order to terminate.

4.4. GMMs and stochastic optimization

There is an interesting link between employing GMMs in the CE setting and other stochastic trajectory optimization based on fixed local 'exploratory' distribution. In case when very few elite samples are available a classical GMM EM algorithm will fail to estimate correct means and covariances. The EM implementation employed in this work was therefore augmented to avoid degeneracies by adding small artificial noise to the estimated covariances after each iteration. As a result even with few samples the algorithm succeeds and could assign as low as a single sample per Gaussian. In such cases the added noise serves the role of the exploratory distribution used to perform local stochastic variations for gradient descent at the next iteration. Another requirement for successful EM is to initialize the components with slightly overlapping covariance ellipsoids so that their union covers the state space of interest.

4.5. Examples

We illustrate the CE method with three examples developed by Kobilarov (2011). The simple car (Figure 1) is an example of parametrization using primitives with constant forward and turning velocities. The resulting trajectories are not constrained to reach the goal; this condition is instead enforced by a penalty term in the cost function. In contrast, the double integrator example (Figure 2) is based on parametrization using states along the path the curves between which can be computed in closed form. Finally, the helicopter example (Figure 3) illustrates a more complex system evolving in 3D with dynamics abstracted using a sequence of trim primitives and maneuvers. The parametrization is computed through inverse kinematics so that all trajectories reach the goal; the cost function is then simply the time of flight.

The car, point mass, and helicopter scenarios take approximately 6, 10, and 30 seconds, respectively, to produce the solutions shown using a PC with Core i7-920XM, 2.0 GHz. The computation is based on rejecting all sampled trajectories that collide with obstacles. For instance, in the helicopter example approximately 85% of all sampled trajectories are rejected. Thus, the computational times can be significantly improved through: (1) parallelized sampling and parallelized collision checking, (2) constructing a probabilistic parametrized trajectory space obstacle model and using it to tilt the CE distribution away from obstacles. Such directions are not explored in this work. Instead, complex constraints such as obstacles will be handled by using a sampling-based motion planning tree as a basis for stochastic optimization, as described next.

5. CE motion planning

We next develop the CE motion planning methodology that combines optimal sampling-based motion planning and global stochastic optimization. These two concepts are linked through a probabilistic model that is being iteratively optimized while at the same time serving as a motion planning sampling distribution. Our particular approach utilizes the RRT* algorithm for planning and the CE method for adaptive sampling. The basic steps are summarized as follows:

Algorithm overview: trajectory-cross-entropy (TCE) motion planning

0. Expand RRT/PRM and attempt to connect to goal region

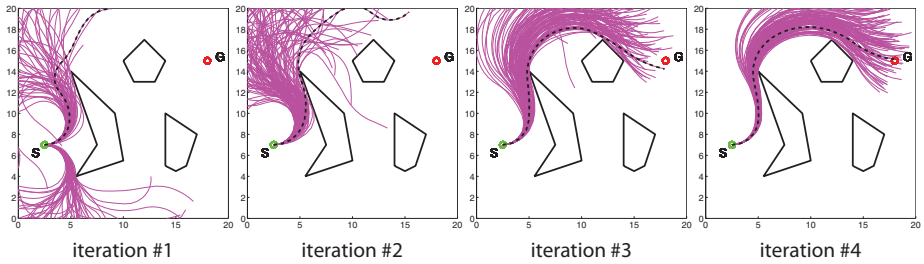


Fig. 1. The first four iterations of the CE algorithm 6 applied to a simple car model. The upper plots show the sampled trajectories $\varphi(Z_1), \ldots, \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). A total of $m = 6$ primitives were used. Iteration #1 shows the resulting set of trajectories after the initialization steps 0.1–0.3 of the CE algorithm.

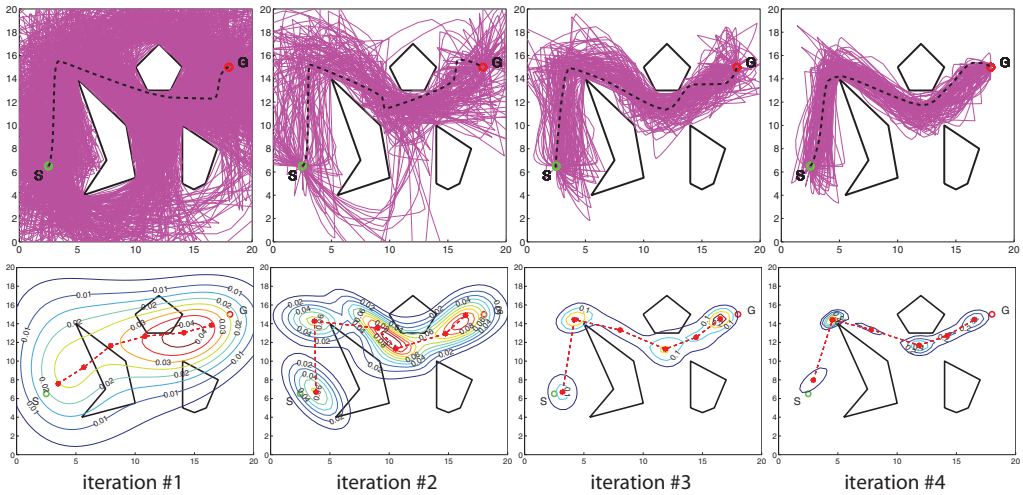


Fig. 2. The first four iterations of the CE algorithm to a double integrator. The upper plots show the sampled trajectories $\varphi(Z_1), \ldots, \varphi(Z_N)$ and the current optimal path $\varphi(z^*)$ (dashed). The lower plots visualize $p(\cdot, \hat{v}_j)$ as the level sets of an induced density over the (x, y)-position space which encodes the lowest cost of trajectories passing through it.

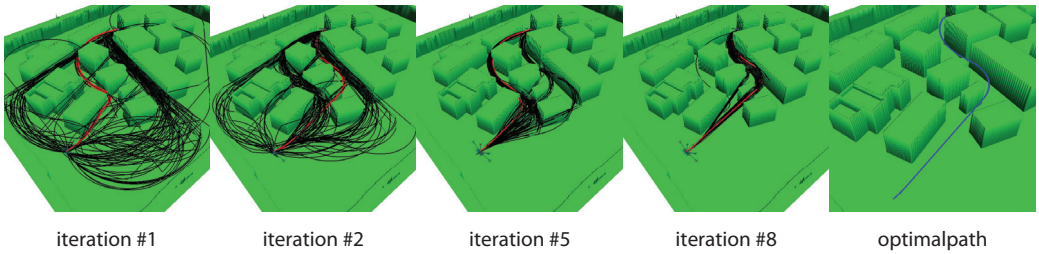


Fig. 3. Several iterations of the CE algorithm applied to the helicopter scenario. As the optimization proceeds the set of samples concentrates in the homotopy classes with minimum trajectory cost.

1. Obtain all RRT/PRM trajectories $\{\pi_i\}_{i=1}^N$ reaching the goal
2. Construct parametrized trajectories $Z_i = \psi(\pi_i)$
3. Update $p_{\mathcal{Z}}$ using the elite subset of these parameters
4. Sample a trajectory $Z \sim p_{\mathcal{Z}}$
5. Select one or more states $X = \varphi(Z, t)$ for a random t and add to RRT/PRM
6. Repeat from either (0) or (1) with some probability. Stop on a termination condition.

The algorithm is based on a CE update of a density $p_{\mathcal{Z}}$ defined over a space of trajectory parameters $z \in \mathcal{Z}$. This density is then used to sample trajectories from which states are extracted and used as RRT/PRM vertices. This can be equivalently regarded as inducing another density on the state space as will be explained below. A simplified version[3] is also developed which bypasses trajectory parametrization and performs density estimation and sampling directly in a state space density $p_{\mathcal{X}}$:

Algorithm overview: state-cross-entropy (SCE) motion planning

0. Expand RRT/PRM and attempt to connect to goal region

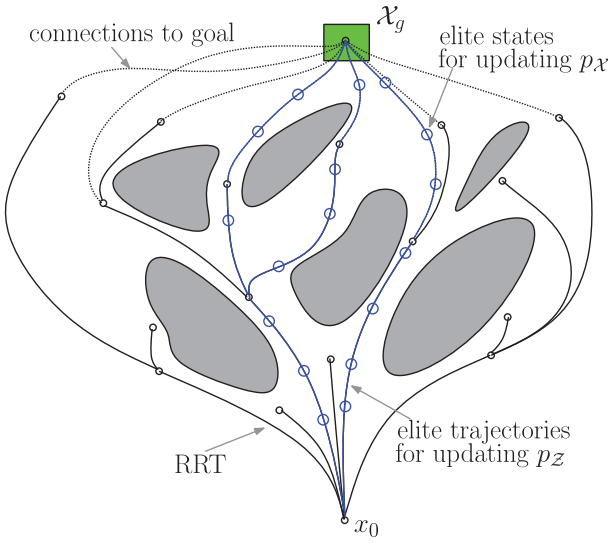


Fig. 4. A motion planning tree with nodes connected to the goal region $\mathcal{X}_g$. Elite trajectories are drawn with thicker lines. Elite states are obtained by discretizing these trajectories, e.g. with a constant time step, and extracting the corresponding states (drawn as circles).

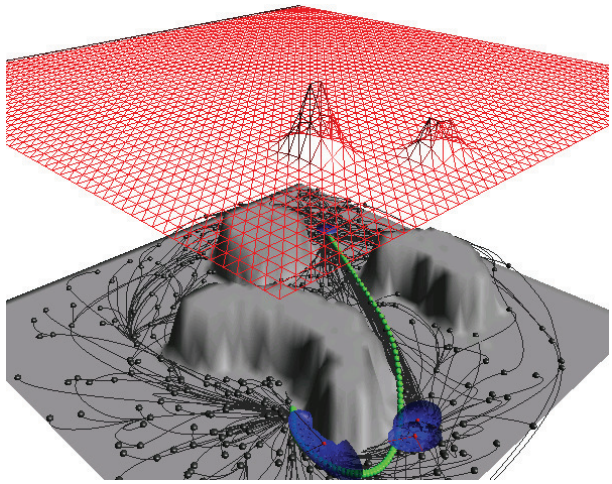


Fig. 5. A tree constructed using the SCE-RRT* algorithm. Gaussian mixture model ellipsoids of the currently updated density are shown. The top graph shows the likelihood $p_{\mathcal{X}}$ (restricted to the planar Euclidean coordinates for better visualization). The peaks identify salient state space regions the are likely to result in low cost trajectories. These regions will be sampled at the next RRT iteration.

1. Obtain all RRT/PRM trajectories $\{\pi_i\}_{i=1}^N$ reaching the goal
2. Discretize each trajectory π_i into a set of states
3. Update $p_{\mathcal{X}}$ using the elite subset of all states of discretized trajectories
4. Sample a state $X \sim p_{\mathcal{X}}$ and add to RRT/PRM
5. Repeat from either (0) or (1) with some probability. Stop on a termination condition.

Figure 4 shows the distinction between using elite states and elite trajectories to update different distributions.

The two methods can be unified by regarding them as optimizations of a special cost function $J(x)$ over the state space $\mathcal{X}$. To make this precise, define the subspace of trajectories $\mathcal{P}_{\text{con}} \subset \mathcal{P}$ which satisfy the constraints and reach the goal, i.e.

$$\mathcal{P}_{\text{con}} = \{(x,u) \in \mathcal{P} \mid \exists T > 0 \text{ s.t. } F(x(t)) > 0 \text{ and } x(T) \in \mathcal{X}_g \text{ for all } t > 0\}.$$

In addition, extend the cost function definition according to $J : \mathcal{P} \cup \mathcal{Z} \cup \mathcal{X} \to \mathbb{R}$ giving the cost of either a trajectory $J(\pi)$, a trajectory parameter $J(z)$, or a single state $J(x)$ depending on the argument. The first two were already defined (see (3) and (16)) by

$$J(\pi) = \int_0^T C(\pi(t))\, dt \quad \text{and} \quad J(z) = \int_0^T C(\varphi(z,t))\, dt,$$

while the new induced cost over $\mathcal{X}$ becomes

$$J(x) = \min_{\pi \in \mathcal{P}_{\text{con}}} \{J(\pi) \mid \exists t \geq 0 \text{ s.t. } \pi(t) = x, \pi(T) \in \mathcal{X}_g, T \geq t\},$$

i.e. the minimum cost over all trajectories that pass through x and reach the goal region. Clearly, we have

$$\min_{x \in \mathcal{X}} J(x) = \min_{\pi \in \mathcal{P}_{\text{con}}} J(\pi) \leq \min_{z \in \mathcal{Z}_{\text{con}}} J(z).$$

From the point of view of stochastic optimization one can regard the problem as either optimizing $J(x)$ using a model $p_{\mathcal{X}}$ (corresponding to the SCE algorithm) or to optimizing $J(\pi)$ using a model $p_{\mathcal{Z}}$ (corresponding to the TCE algorithm). In both cases the data is provided using a motion planning method such as RRT*. The motion planning method uses a sampling density which in SCE case is precisely the same $p_{\mathcal{X}}$, while in the TCE case is a distribution on $\mathcal{X}$ that is implicitly induced by $p_{\mathcal{Z}}$. We next detail the resulting algorithms based on these ideas.

5.1. The SCE RRT* algorithm

The SCE method is a straightforward extension to the modified RRT* algorithm outlined in Section 3.2. It is implemented as a special `Sample` function (Algorithm 7) which has access to the set of tree nodes reaching the goal. This function is called from the main `RRT*` routine (Algorithm 1).

The function performs CE sampling with probability r_{ce}, otherwise reverts to standard uniform sampling over the space of interest in $\mathcal{X}$, denoted by $X \sim \texttt{Uniform}(\mathcal{X})$. The function $\tau_{\min}(\mathcal{N}_g)$ returns the smallest time among the nodes $\mathcal{N}_g$. This time is used to determine the time step h with which trajectories reaching the goal will be quantized to extract states for updating the sampling distribution. The minimum required number of such states $N_{\min}^{\mathcal{X}}$ should either result in at least $2n$ elite samples (where $n = \dim(\mathcal{X})$) or

Algorithm 7: $X \leftarrow \texttt{Sample}(i, \mathcal{N}_g)$

parameters: r_{ce}–CE sampling ratio, m–path discretization, k–GMM components

1 **if** $\texttt{rand}() < r_{ce}$ **then**
2 $\quad N^{\mathcal{X}}_{\min} = \texttt{max}(2n/\rho, 2nk)$
3 $\quad h = \tau_{\min}(\mathcal{N}_g)/m$
4 $\quad X \leftarrow \texttt{SCE\_Sample}(\mathcal{N}_g, h, N^{\mathcal{X}}_{\min})$
5 $\quad$ **if** $X \neq \emptyset$ **then**
$\quad\quad$ **return** X
repeat
6 $\quad$ sample $X \sim \texttt{Uniform}(\mathcal{X})$
until $\texttt{ObstacleFree}(X)$
7 **return** X

Algorithm 8: $X \leftarrow \texttt{SCE\_Sample}(\mathcal{N}_g, h, N_{\min})$

1 $\mathfrak{X} \leftarrow \emptyset$
2 **foreach** $\eta_g \in \mathcal{N}_g$ **do**
3 $\quad$ backtrack path π connecting η_0 and η_g
4 $\quad$ **for** $t = h;\ t < \tau(\pi);\ t = t + h$ **do**
5 $\quad\quad \mathfrak{X} \leftarrow \{\mathfrak{X}, (\pi_x(t), \texttt{CostToCome}(\eta_g))\}$
6 **if** $|\mathfrak{X}| > N_{min}$ **then**
7 $\quad p_{\mathcal{X}} \leftarrow \texttt{CE\_Estimate}(\mathfrak{X})$
$\quad$ **repeat**
8 $\quad\quad$ sample $X \sim p_{\mathcal{X}}$
$\quad$ **until** $\texttt{ObstacleFree}(X)$
$\quad$ **return** X
else
$\quad$ **return** $\emptyset$

Algorithm 9: $p^* \leftarrow \texttt{CE\_Estimate}\left(\{Z_i, \gamma_i\}_{i=1}^N\right)$

parameters: ρ—fraction of elite samples

1 $\gamma \leftarrow \texttt{Quantile}(\{\gamma_i\}_{i=1}^N, \rho)$
2 $\hat{v}^* \leftarrow \underset{v \in \mathcal{V}}{\text{argmax}} \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} I_{\{\gamma_i < \gamma\}} \ln p(Z_i, v)$
3 **return** $p(\cdot; \hat{v}^*)$

be proportional to the required number of GMM components. Note that these are guidelines that have resulted in good performance in practice but other choices for $N^{\mathcal{X}}_{\min}$ are possible as well.

Once the list of states and the costs of optimal trajectories passing through them have been assembled in the set $\mathfrak{X}$, it is used to update the distribution using a generic `CE_Estimate` routine (Algorithm 9). Note that this routine will be used for updating both distributions over $\mathcal{X}$ and $\mathcal{Z}$ but for generality is defined on the space $\mathcal{Z}$.

An illustration of the density $p_{\mathcal{X}}$ and its likelihood is given in Figure 5 for a double integrator system with

Algorithm 10: $X \leftarrow \texttt{Sample}(i, \mathcal{N}_g)$

parameters: r_{ce}—CE sampling ratio, m—path discretization, k—GMM component

1 **if** $\texttt{rand}() < r_{ce}$ **then**
2 $\quad N^{\mathcal{Z}}_{\min} = 2mk$
3 $\quad X \leftarrow \texttt{TCE\_Sample}(\mathcal{N}_g, N^{\mathcal{Z}}_{\min})$
4 $\quad$ **if** $X = \emptyset$ **then**
5 $\quad\quad N^{\mathcal{X}}_{\min} = max(2n/\rho, 2nk)$
6 $\quad\quad h = \tau_{\min}(\mathcal{N}_g)/m$
7 $\quad\quad X \leftarrow \texttt{SCE\_Sample}(\mathcal{N}_g, h, N^{\mathcal{X}}_{\min})$
8 $\quad$ **if** $X \neq \emptyset$ **then**
$\quad\quad$ **return** X
repeat
9 $\quad$ sample $X \sim \texttt{Normal}(\mathcal{X})$
until $\texttt{ObstacleFree}(X)$
10 **return** X

bounded acceleration (Karaman et al. 2011) moving optimally in a natural terrain. Further details and empirical analysis are given in Section 6.

5.2. The TCE RRT* algorithm

The motivation behind the TCE method is to fully exploit not only individual states in the state space but also correlation between states along whole trajectories. Search over parametrized trajectories has proven key in the stochastic optimization techniques (see Sections 1 and 4.2). Yet, as explained in Section 4 the basic CE method does not easily extend to highly constrained settings. The proposed TCE algorithm overcomes these issues by integrating the CE trajectory optimization (Section 4) with RRT* for handling complex environments.

The TCE algorithm is implemented through the `Sample` (Algorithm 10) function called from the main `RRT*` routine (Algorithm 1). If not enough data is available then the algorithm reverts to SCE sampling (line 4). The parametrization we choose is based on a finite set of states along the path connected with optimal trajectories (as explained in Section 4.1), i.e. $\mathcal{Z} = \mathcal{X}^m$ and

$$z = (x_1, \ldots, x_m),$$

where m is a chosen number. The parametrization then becomes

$$\varphi_x(z) = (x_0, x_1)(x_1, x_2) \cdots (x_{m-2}, x_{m-1}) \\ (x_{m-1}, x_m), (x_m, \mathcal{X}_g)$$

where (x_a, x_b) is an optimal obstacle-free trajectory between two states x_a and x_b computed using the `Steer` command defined in Section 3.2. The function extracting parameters from a given trajectory $\pi = (x, u)$ is then

$$\psi(\pi) = (x(h), x(2h), \ldots, x(mh)),$$

Algorithm 11: $X \leftarrow$ `TCE_Sample`$(\mathcal{N}_g, N_{\min})$

$\mathfrak{Z} \leftarrow \emptyset$
foreach $\eta_g \in \mathcal{N}_g$ **do**
  backtrack path π connecting η_0 and η_g
  convert to parameters $Z \leftarrow \psi(\pi)$
  $\mathfrak{Z} \leftarrow \{\mathfrak{Z}, (Z, \texttt{CostToCome}(\eta_g))\}$
if $|\mathfrak{Z}| < N_{min}$ **then**
  return $\emptyset$
else
  $p_{\mathcal{Z}} \leftarrow \texttt{CE\_Estimate}(\mathfrak{Z})$
repeat
  sample $Z \sim p_{\mathcal{Z}}$
  sample $t \sim \texttt{Uniform}([0, \tau(Z)])$
  $X \leftarrow \varphi(Z, t)$
until `ObstacleFree`(X)
return X

where $h = T/(m+1)$ and T is the shortest time among all trajectories ending in $\mathcal{N}_g$.

This representation was chosen since it naturally fits into the RRT framework through the edge creation process. It is also meaningful considering the L_2 metric on trajectories since

$$\int_0^T \|x_a(t) - x_b(t)\|^2 \, dt \approx \sum_{i=1}^{m} \|x_a(ih) - x_b(ih)\|^2 = \|z_a - z_b\|^2, \tag{19}$$

which approaches equality as $m \to \infty$. This is also the type of metric employed by parametric density estimators such as EM. On the other hand, other metrics such as L_∞ (see e.g. Yershov and LaValle 2011) or Hausdorff (Knepper et al. 2010) distances can also be considered in a more general, i.e. non-parametric setting.

With such a choice of parametrization the algorithm extracts trajectories reaching the goal, converts them into parameter vectors, and updates the distribution using their elite subset. A new trajectory is then sampled and a state along that trajectory is selected to join the tree as long as the whole trajectory is free of obstacles.

The resulting density can be visualized in Figure 6. The density $p_{\mathcal{Z}}$ is difficult to display so an approximation of the 'induced density' $p_{\mathcal{X}}$ is used, formally defined by

$$p_{\mathcal{X}}(X) = \eta \cdot \max_{Z \in \mathcal{Z}_{\text{con}}} \{p_{\mathcal{Z}}(Z) \mid X = \varphi_x(Z, t) \text{ for some } 0 < t < \tau(Z)\}, \tag{20}$$

where $\eta > 0$ is a normalizing constant. The approximate density shown in Figure 6 was built by partitioning the subspace of planar Euclidean coordinates into a grid, drawing 10,000 trajectory samples from $p_{\mathcal{Z}}$, tracing the corresponding trajectories and retaining the maximum likelihood value in each traced cell. The unvisited cells are set to 0. Further examples and empirical analysis of the algorithm will be given in Section 6.

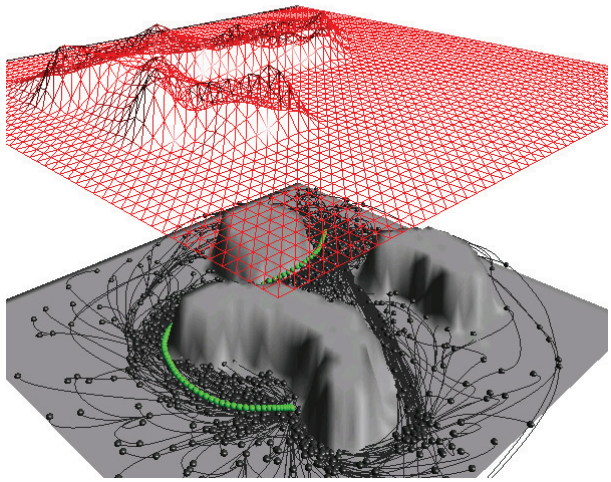


Fig. 6. An RRT* built using TCE sampling. The top graph visualizes the likelihood $p_{\mathcal{Z}}$ transformed into another likelihood $p_{\mathcal{X}}$ that encodes the cost of the optimal trajectory passing through a given state. Although only three Gaussian mixture components in trajectory parameter space $\mathcal{Z}$ were used, the induced likelihood in state space $\mathcal{X}$ has a non-trivial landscape. It corresponds to salient states that are likely to be sampled in the next iteration.

5.3. User parameters

The algorithm depends on several user-defined parameters that require further investigation (Table 1). The elite fraction ρ typically require minor tuning since the listed default values have proven effective for various classes of problems (Rubenstein and Kroese 2008). We set the CE sampling ratio r_{ce} to 0.5 in order to balance equally between exploring the space and exploiting the collected information about trajectory costs. In the absence of enough information the algorithm automatically reverts to exploration. This parameter can be optimized on-line using more sophisticated techniques. Path discretization parameter m determines how much information is extracted from each trajectory. A very fine discretization results in redundant information that is unnecessary and will not improve the density estimates. On the other hand, very complicated maze-like environments would require long trajectories and more segments. A rule of thumb is to keep track of the average number of tree edges along paths reaching the goal and set m to be at least as high. The optimal number of GMM components is problem dependent and currently no general procedure is available to select k. In theory, each component should 'learn' the most promising state space regions or, respectively, trajectory homotopy classes. In practice, for the examples studied in this work more than four components did not improve the resulting trajectory cost. This could also be explained through the fact that it is non-trivial

Table 1. User-defined parameters affecting the algorithms.

Description	Variable	Range	Default value
Elite fraction	ρ	$[0.01, 0.1]$	0.1
CE sampling ratio	r_{ce}	$[0, 1]$	.5
Path discretization	m	> 0	8
GMM components	k	> 0	4

CE, cross-entropy; GMM, Gaussian mixture model.

to populate all homotopy classes of trajectories reaching the goal even with fast exploration methods such as RRT.

5.4. Limitations

The proposed CE motion planning methods are applicable to problems for which a single or multiple paths to the goal can be computed during the algorithm execution. If only one path was computed then the proposed approach behaves similarly to existing local stochastic optimization techniques. The greater the number of paths that become available, the better the model that can be constructed and exploited for further sampling. The GMM probabilistic model was chosen for computational convenience and due to empirical evidence as well as links to other successful methods. Precise guidelines for selecting a representation that optimally trades off efficiency and optimality are still lacking. In this context, it would be also useful to consider non-parametric representations such as locally weighted learning techniques (Atkeson et al. 1997) due to their incremental nature and computational efficiency, or sparse Gaussian process (GP) classification (Rasmussen and Williams 2005) with available formal bounds (Seeger 2003) that could be useful for establishing performance guarantees. Another issue that could effect performance and requires further study is the relation between the chosen model and the environment complexity. Finally, the proposed methods are useful in higher dimensions where dense sampling is computationally intractable. In problems with up to a few dimensions we expect that standard methods can simply be run longer to obtain comparable results.

6. Examples

We use two simple simulated examples to test the proposed methods. They correspond to commonly used models and were chosen since optimal steering procedures required by the RRT algorithms are readily available.

6.1. Double integrator

We consider a double integrator system moving in a constrained three-dimensional environment. This is a higher-dimensional version of the example considered by Karaman et al. (2011). The system has state space $\mathcal{X} = \mathbb{R}^3 \times \mathbb{R}^3$ with state $x = (q, v)$ consisting of the position q and velocity v.

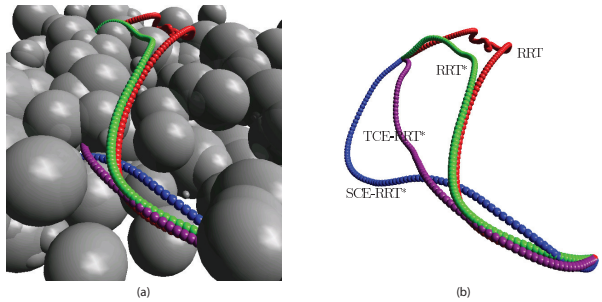


Fig. 7. (a) A box environment with randomly placed spheres and computed optimal trajectories by the four methods using 5,000 samples. (b) The same trajectories with obstacles removed for clearer view. The resulting trajectory costs are: RRT, 21.09; RRT*, 13.73; SCE-RRT*, 11.39; TCE-RRT*, 10.70. While the random sampling seed is identical for all four methods, the optimal solution computed by the three RRT* methods lie in different homotopy classes. The CE computations were performed with $m = 8$ for trajectory quantization and $k = 4$ Gaussian mixture components.

The control space $\mathcal{U} \subset \mathbb{R}^3$ consists of the accelerations u. The dynamics is

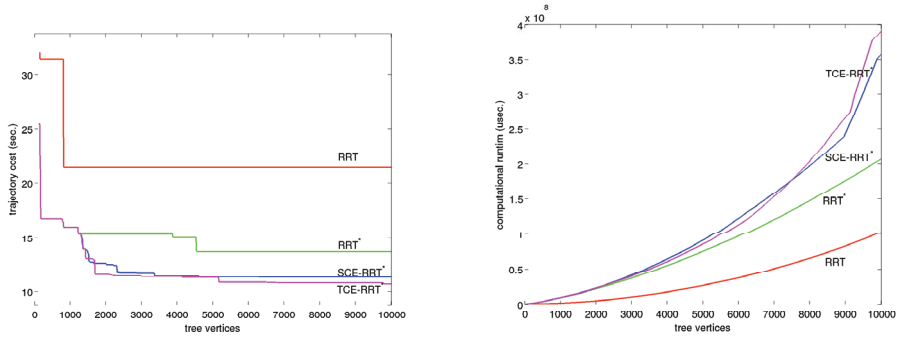
$$\dot{q} = v, \quad \dot{v} = u, \quad \|u\| < u_{max},$$

where $u_{\max} > 0$ is a given scalar bound. We consider the time-optimal planning problem, i.e. the cost defined in (3) is

$$C(x, u) = 1.$$

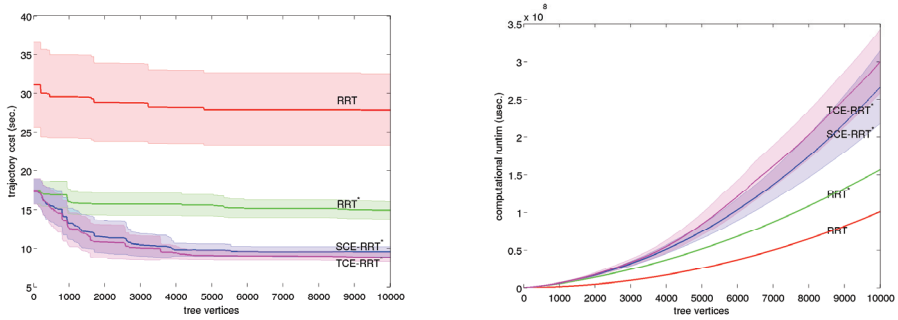
The $\texttt{Steer}(x_a, x_b)$ function computes the time-optimal trajectory between the two states. A time-optimal profile of a one-dimensional double integrator can be computed in closed form since it consists of two intervals of constant acceleration. To extend to multiple dimensions the time-optimal acceleration profiles are computed for each dimension, the one with highest resulting time is retained, and the other profiles are then recomputed under that final time constraint. This is accomplished directly through the solution of quadratic equations in closed form with some bookkeeping.

We construct an environment bounded by a box in three dimensions, populated with spheres with random centers inside the box and random radiuses of up to half the smallest dimension of the box. Figure 7 shows an example box environment with dimensions $50 \times 50 \times 10$ meters populated with 300 spheres. The spheres can intersect and create non-trivial concave obstacles and multiple homotopy classes of paths. We study the performance of four sampling-based methods: the standard RRT (LaValle 2006); the RRT* (Karaman et al. 2011); and the two adaptive sampling methods proposed in this work abbreviated by SCE-RRT* (Section 5.1) and TCE-RRT* (Section 5.2). The RRT* algorithms were constructed using nearest neighbor set with size $|\mathcal{N}_{\text{near}}| = \gamma \log(|\mathcal{T}|)$. We set $\gamma = 10$ since in our experiments smaller values resulted in poorer solutions by all three RRT*-based methods.



(a) (b)

Fig. 8. Computation details for the scenario in Figure 7: (a) resulting trajectory costs as a function of vertices; (b) computational time taken as the number of vertices increases.



(a) (b)

Fig. 9. Double integrator trajectory costs (a) and computational times (b) computed by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT*. Results are averaged over 20 Monte Carlo runs.

Even though the double integrator is a simple system, the devised six-dimensional scenario cluttered with obstacles is a challenging planning problem. As our results illustrate sampling-based methods including RRT* can greatly benefit from an informed adaptive sampling. The proposed CE sampling methods compute lower-cost solutions quicker (see Figure 8). The required computational effort is greater than existing methods but is offset by the ability to find more superior solutions using a smaller number of nodes. Note that these are only empirical observations. Establishing convergence rates and sample complexity formally remains an open problem. Figure 9 provides averaged results from multiple randomized runs.

6.2. Simple air vehicle

We employ the simple fixed wing vehicle model (Karaman and Frazzoli 2010) consisting of decoupled models of the Dubins car in the plane and a double integrator in altitude. The state space is $\mathcal{X} = \mathrm{SE}(2) \times \mathbb{R}^2$ with state $x = (\theta, x, y, z, v_z)$. The control space is $\mathcal{U} \subset \mathbb{R}^2$ with controls $u = (\omega, a_z)$ such that $|\omega| < \tan(\phi_{\max})$ and $|a_z| \leq a_{\max}$, where $\phi_{\max}$ is the maximum steering angle and $a_{\max}$ is the maximum altitude acceleration. The vehicle dynamics is defined according to

$$\dot{\theta} = \omega, \quad \dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{z} = v_z, \quad \dot{v}_z = a_z,$$

where $v > 0$ is the constant forward velocity. We are interested in time optimal motions. Following Karaman and Frazzoli (2010), we employ a steering procedure which initially computes decoupled optimal Dubins curves in the plane and optimal double integrator curve in altitude. If the double integrator curve takes less time than Dubins, then it is recomputed using the Dubins time as a final time constraint. Otherwise the Dubins velocities and times are scaled to match the longer time required by the double integrator.

Density estimation is performed in the ambient space of $\mathcal{X}$ (in the SCE case) and $\mathcal{Z}$ (in the TCE case). For instance, on $\mathcal{X} = \mathrm{SE}(2) \times \mathbb{R}^2 \sim S^1 \times \mathbb{R}^2 \times \mathbb{R}^2$, the angle θ is regarded as a vector so that $x := (\cos(\theta), \sin(\theta), x, y, z, v_z)$ and the CE update is performed in $\mathbb{R}^6$ without internally enforcing $(x^1, x^2) \in S^1$. Once the estimation (i.e. using EM) completes the estimated vectors $(\hat{x}^1, \hat{x}^2)$ are projected onto the circle. This corresponds to using the von Mises distribution to approximate a Gaussian on the circle. A more generally applicable approach is to consider direct estimation on $\mathrm{SE}(2)$ using Lie group methods (Chirikjian 2012),

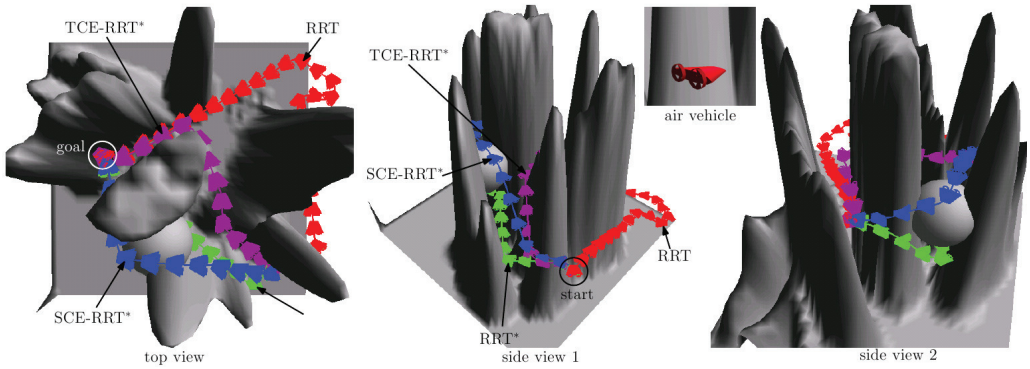


Fig. 10. Air vehicle paths through a digital terrain, computed by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT* using 1,000 vertices. Only a few states are rendered along the paths for better visibility.

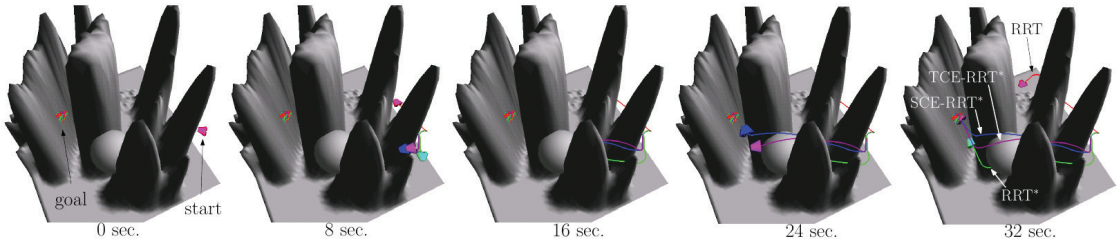


Fig. 11. Snapshots in time of the solution trajectories computed by the four methods. TCE-RRT* and SCE-RRT* compute a 32 second path; the RRT* path lasts 37 seconds; RRT takes 47 seconds to reach the goal.

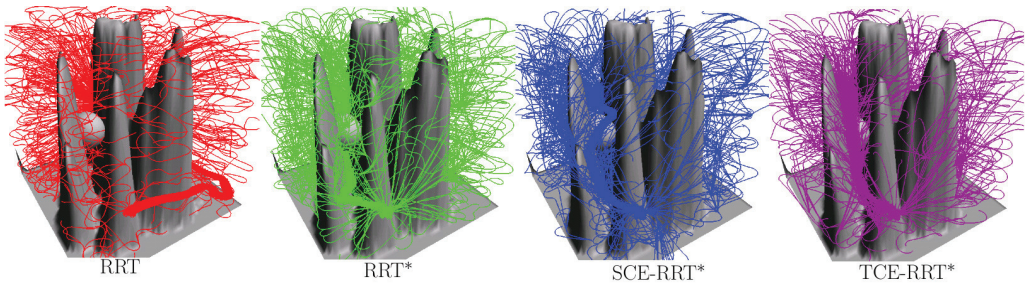


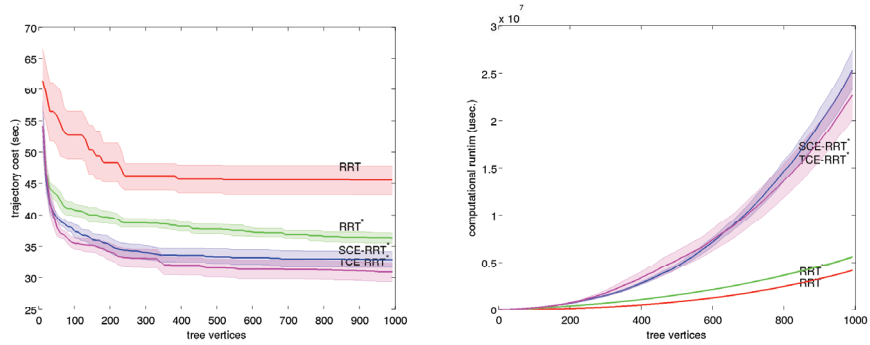
Fig. 12. Roadmaps with 1,000 vertices used to compute the trajectories shown in Figure 11.

which would be especially useful in higher dimensions, e.g. for systems consisting of rigid bodies.

A scenario with uneven terrain depicted in Figure 10 is used to compare the performance of the proposed adaptive sampling methods to RRT and RRT*. In this example we used $k = 4$ Gaussian components and $m = 8$ discrete trajectory states. Figure 11 shows one particular solution to illustrate the type of paths computed by each method. Results from 20 Monte Carlo runs are shown on Figure 13. The CE sampling methods are able to reduce the computed costs even after a few hundred iterations. Yet, as the number of iterations increases their required computational time becomes an order of magnitude higher than RRT and RRT*. This is because the number of trajectories reaching the goal (and, hence, the accumulated information about trajectory costs) is increasing and requires more processing. This is a drawback of our basic CE implementation based on a fixed quantile and iterative GMM processing. The issue can be remedied though for instance by employing incremental, localized, and/or sparsified models as discussed in Section 5.4.

7. Conclusion

In this paper we have proposed a methodology for improving the performance of sampling-based motion planning, i.e. for computing trajectories with lower costs more efficiently. The key point is to use a probabilistic model in either state space and parametrized trajectory space that



(a) (b)

Fig. 13. Air vehicle trajectory cost (a) and computational times (b) using the environment depicted in Figure 10 taken by the four algorithms: RRT, RRT*, SCE-RRT*, TCE-RRT*. Results are averaged over 20 Monte Carlo runs.

is estimated based on the cost of generated paths reaching the desired goal region. The CE method for stochastic optimization is used to adapt the model towards regions of progressively lower cost. The approach is coupled with the optimal rapidly-exploring random tree (RRT*) which uses the model as a sampling distribution and at the same time updates it with newly explored trajectories. Empirical evidence suggests that such adaptive sampling produces lower cost solutions with fewer iterations. Future work will evaluate model representations in view of system and environment complexity and the resulting trade-off between efficiency and optimality. Establishing formal convergence guarantees and sample complexity is another important aspect that must be addressed.

Notes

1. The density q dominates Hp when $q(z) = 0 \Rightarrow H(z)\,p(z) = 0$.
2. The author thanks C. Atkeson for pointing out the closely related CMA method.
3. The author thanks A. Bagnell for a discussion that lead to including this version of the algorithm.

Funding

The author was partially supported by the Keck Institute for Space Studies, Caltech.

Acknowledgements

The author thanks the reviewers for the useful directions for improving the paper.

References

Atkeson CG, Moore AW and Schaal S (1997) Locally weighted learning. *Artificial Intelligence Review* 11: 11–73.

Botev Z and Kroese DP (2004) Global likelihood optimization via the cross-entropy method with an application to mixture models. In *Winter Simulation Conference*, pp. 529–535.

Burns B and Brock O (2005a) Sampling-based motion planning using predictive models. In *IEEE International Conference on Robotics and Automation*, April 2005, pp. 3120–3125.

Burns B and Brock O (2005b) Toward optimal configuration space sampling. In *Robotics: Science and Systems*.

Chirikjian GS (2012) Applied and numerical harmonic analysis. *Stochastic Models, Information Theory, and Lie Groups. Volume II: Analytic Methods and Modern Applications*. Basel: Birkhäuser.

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

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

Costa A, Jones OD and Kroese D (2007) Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters* 35: 573–580.

De Bonet JS, Isbell CL Jr and Viola PA (1996) Mimic: Finding optima by estimating probability densities. In *Proceedings of NIPS'96*, pp. 424–430.

Denny J and Amato NM (2011) Toggle PRM: Simultaneous mapping of C-free and C-obstacle - a study in 2D. In *Proceedings IEEE IROS2011*, pp. 2632–2639.

Figueiredo MAF and Jain AK (2002) Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24: 381–396.

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

Homem-de Mello T (2007) A study on the cross-entropy method for rare-event probability estimation. *INFORMS Journal on Computing* 19: 381–394.

Hsu D, Latombe J-C and Kurniawati H (2006) On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research* 25: 627–643.

Hsu D, Sanchez-Ante G and Sun Z (2005) Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA 2005)*, pp. 3874–3880.

Hu J, Fu MC and Marcus SI (2007) A model reference adaptive search method for global optimization. *Operations Research* 55: 549–568.

Hu J and Hu P (2009) On the performance of the cross-entropy method. In *Winter Simulation Conference (WSC '09)*, pp. 459–468.

Igel C, Suttorp T and Hansen N (2006) A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation GECCO*. New York: ACM Press, pp. 453–460.

Kalakrishnan M, Chitta S, Theodorou E, Pastor P and Schaal S (2011) STOMP: stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Kalisiak M and van de Panne M (2007) Faster motion planning using learned local viability models. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'07)*, pp. 2700–2705.

Karaman S and Frazzoli E (2010) Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (to appear)*, 2011.

Karaman S, Walter MR, Perez A, Frazzoli E and Teller S. Anytime motion planning using the RRT*. In *IEEE Conference on Robotics and Automation (ICRA)*, April 2011.

Knepper RA and Mason MT (2011) Realtime informed path sampling for motion planning search. In *15th International Symposium on Robotics Research (ISRR)*.

Knepper RA, Srinivasa S and Mason MT (2010) An equivalence relation for local path sets. In Latombe J-C, Hsu D, Isler V and Lin MC (eds), *The Ninth International Workshop on the Algorithmic Foundations of Robotics*. Heidelberg: Springer.

Kobilarov M (2011) Cross-entropy randomized motion planning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, June 2011.

Kroese D, Porotsky S and Rubinstein R (2006) The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability* 8: 383–407.

Kurniawati H and Hsu D (2004) Workspace importance sampling for probabilistic roadmap planning. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, vol. 2, pp. 1618–1623.

Ladd AM and Kavraki LE (2005) *Fast Tree-Based Exploration of State Space for Robots with Dynamics*. Berlin: Springer, pp. 297–312.

Larrañaga P and Lozano JA (eds) (2002) *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Dordrecht: Kluwer Academic Publishers.

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

Li Y and Bekris KE (2010) Balancing state-space coverage in planning with dynamics. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3246–3253.

Margolin L (2005) On the convergence of the cross-entropy method. *Annals of Operations Research* 134: 201–214.

McLachlan G and Peel D (2002) *Finite Mixture Models*. New York: John Wiley & Sons, Inc.

Moore A and Schneider J (1995) Memory-based stochastic optimization. In *Neural Information Processing Systems 8*.

Murray RM, Rathinam M and Sluis WM (1995) Differential flatness of mechanical control systems. In *Proceedings ASME International Congress and Exposition*.

Murray RM and Sastry S (1993) Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control* 38: 700–716.

Pelikan M, Goldberg DE and Lobo FG (2002) A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications* 21: 5–20.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (*Wiley Series in Probability and Statistics*). New York: John Wiley & Sons, Inc.

Rasmussen CE and Williams CKI (2005) *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA: The MIT Press.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'09)*, pp. 489–494.

Rickert M, Brock O and Knoll A (2008) Balancing exploration and exploitation in motion planning. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 2812–2817.

Rubinstein RY and Kroese DP (2004) *The Cross-entropy Method: A Unified Approach to Combinatorial Optimization*. New York: Springer.

Rubenstein RY and Kroese DP (2008) *Simulation and the Monte Carlo Method*. New york: John Wiley & Sons, Inc.

Seeger M (2003) Pac–Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research* 3: 233–269.

Spall JC (2003) *Introduction to Stochastic Search and Optimization*. New York: John Wiley & Sons, Inc..

Theodorou E, Buchli J and Schaal S (2010) A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research* 11: 3137–3181.

Yershov D and LaValle S (2011) Sufficient conditions for the existence of resolution complete planning algorithms. In Hsu D, Isler V, Latombe J-C and Lin M (eds), *Algorithmic Foundations of Robotics IX* (*Springer Tracts in Advanced Robotics*, vol. 68). Berlin: Springer, pp. 303–320.

Zhigljavsky A and Zilinskas A (2008) *Stochastic Global Optimization*. New York: Springer.

Zucker M, Kuffner J and Bagnell JA (2008) Adaptive workspace biasing for sampling-based planners. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA'08)*, pp. 3757–3762.