The Importance of a Suitable Distance Function in Belief-Space Planning

Zakary Littlefield, Dimitri Klimenko, Hanna Kurniawati and Kostas E. Bekris

1 Introduction

Uncertainty is ubiquitous, since sensing is never perfect, actuators have errors, and a robot's operating environment is often unknown. Due to this imperfect information and errors, the exact robot state is never perfectly known. Therefore, instead of finding an optimal solution in the state space, many methods represent uncertainty about the robot state as a probability distribution, and plan in the set of distributions over states, called the belief space [12, 24, 25, 31]. The computational complexity of planning in the belief space is much higher than in the state space because the size of the belief space is doubly exponential in the number of state space dimensions. Nevertheless, recent advances have shown that motion planning in belief space is becoming practical for many medium size problems [1, 6, 9, 32].

Background: Interestingly, progress in belief-space planning has been achieved through similar tools to those used in the deterministic case. In particular, this progress was achieved by sampling a small set of representative beliefs and planning with

Zakary Littlefield is supported by the NASA Space Technology Research Fellowship (NNX13AL71H). Dimitri Klimenko is supported by the Australian Postgraduate Award. Work by the Rutgers authors has been supported by NSF CCF:13307893, NSF IIS:1451737.

Z. Littlefield · K.E. Bekris (✉)
Computer Science Department, Rutgers University, New Brunswick, NJ, USA
e-mail: zwl2@cs.rutgers.edu

K.E. Bekris
e-mail: kostas.bekris@cs.rutgers.edu

D. Klimenko · H. Kurniawati
School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia
e-mail: dimitri.klimenko@uqconnect.edu.au

H. Kurniawati
e-mail: hannakur@uq.edu.au

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_39

respect to only this small set of sampled beliefs. In fact, many methods (e.g., [3, 18, 25]) in belief-space planning are extensions of sampling-based ones for the deterministic case, such as PRM, RRT, PRM*, and RRG [4, 7]. These methods typically restrict beliefs to be represented by Gaussian parameters or consider the maximum likelihood estimate of the state.

Recent work, which has shown that one can achieve asymptotic optimality without a steering function in the deterministic case [15, 20], has the potential to allow an even more straightforward way to extend sampling-based planners to belief-space planning. The similarity between deterministic planning without a steering function and belief-space planning indicate that properties critical for deterministic motion planning are likely to be critical for belief-space motion planning as well.

Similar to sampling-based methods for the deterministic case, many equivalent approaches for belief-space planning rely on distances between beliefs to partially guide their sampling and pruning operations [8, 12, 29]. Many distance functions can be potentially used and they can have significantly different effects in belief-space planning. Nevertheless, the effectiveness of the different distance functions has not been studied in the related literature on belief-space planning.

This paper focuses on understanding the suitability of commonly used distance functions in belief-space motion planning. Commonly used functions, such as L1 and Kullback–Leibler divergence KL, in general ignore the underlying distance in the state space. As a result, two beliefs whose supports do not overlap, but lie near each other in the state space, will have the same distance as two beliefs whose supports lie very far away. Figure 1 illustrates this issue. If the supports of the beliefs are unbounded, the above problems are less severe, though they still exist. While the Wasserstein or Earth Mover's Distance (EMD) alleviates the aforementioned issue, it has been rarely used in the related literature [11, 14]. This paper presents a comparative study of the effect of these metrics in two classes of belief space planning, i.e., Non-Observable Markov Decision Processes (NOMDPs) and Partially Observable Markov Decision Processes (POMDPs).

NOMDP Evaluation: NOMDP is the simplest class of belief-space planning, a planning under uncertainty challenge where no observation is available. Despite its simplicity, NOMDP has often been applied as an intermediate solution to complex planning under uncertainty problems [10]. The simplicity of this class of problems allow us to compare the metric on various complex motion planning problems, which are still unsolvable when the challenge is partially observable. To solve NOMDPs, this

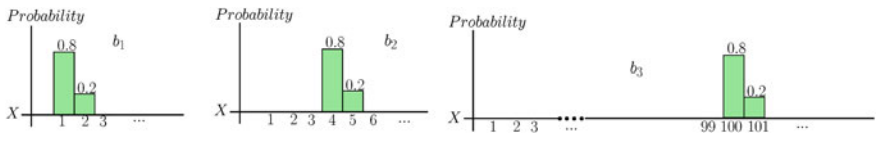


Fig. 1 Illustration of the problem with L1 and KL. Suppose the state space $\mathbb{X}$ is the 1-dim. Natural numbers and the distance $d_{\mathbb{X}}(x, x') = |x - x'|$. Then, the L1 distance $D_{L1}(b_1, b_2) = D_{L1}(b_1, b_3)$, the KL distance $D_{KL}(b_1, b_2) = D_{KL}(b_1, b_3)$, and the EMD distance $D_W(b_1, b_2) < D_W(b_1, b_3)$

paper leverages recent results in deterministic motion planning that show asymptotically optimal solutions [7] can be computed by sampling-based methods that do not require steering functions [15, 20]. These methods are extended to solve NOMDP problems. Such extensions are simpler compared to extending methods that require a steering function, because computing a good steering function in the belief space is non-trivial. Simulation results indicate that as the NOMDP problem becomes more complex, the differences in the effectiveness of different distance functions become quite prominent. In fact, in state spaces with more than 4 dimensions, just by replacing L1 or KL distance with EMD, the speed of solving the problems improves substantially and the problems transition from virtually unsolvable to solvable.

POMDP Evaluation: The second class of problems in our comparative study is the Partially Observable Markov Decision Processes (POMDP). To solve POMDP problems, Monte Carlo Value Iteration (MCVI) [2] is used here, which is an offline POMDP-solver designed for problems with continuous state spaces. In this paper, we only apply the various metrics to a 2D navigation problem when evaluating the performance in the POMDP framework, due to the limitation of existing POMDP solvers in solving problems with large action spaces. Although this limitation means we cannot yet show the full potential of EMD in POMDP, the preliminary result reveals that EMD could significantly reduce the number of belief-space samples that sampling-based POMDP-solvers need to reach a certain solution quality.

Overall Contributions: The results of this comparative study indicate that EMD is more suitable than L1 and KL, and could significantly improve the performance of belief space planning, even though its computation can be computationally more expensive. Steps towards the efficient computation of the EMD are also described here. Furthermore, this paper shows that EMD carries the Lipschitz continuity of the cost function in the state space to Lipschitz continuity of expected cost in the belief space. This is useful because this property is used in the convergence analysis of several asymptotically optimal motion planning methods without a steering function [15, 20], including methods for belief-space planning [12, 13].

2 Problem Setup for Comparative Study

A POMDP is a mathematically principled framework for planning under uncertainty in discrete-time. Formally, a POMDP is defined as a tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$, where S is the set of states, A is the set of actions, and O is the set of observations. The notation T represents motion uncertainty and is defined as a conditional probability function $T(s, a, s') = \mathbb{P}(s'|s, a)$, where $s, s' \in S$ and $a \in A$. The notation Z represents sensing uncertainty and is defined as a conditional probability function $Z(s', a, o) = \mathbb{P}(o|s', a)$, where $s' \in S$, $a \in A$, and $o \in O$. At each step, a POMDP agent is in a state $s \in S$, takes an action $a \in A$, moves from s to an end state $s' \in S$, perceives an observation $o \in O$, and receives a reward $R(s, a)$ for taking action a from state s. However, a POMDP agent never knows this exact state, and instead reasons with respect to distributions over states, called beliefs. At each step, the agent's

belief estimate is updated based on the action it just performed and the observation perceived. The agent's goal is to choose a suitable sequence of actions that will maximize its expected total reward, when the agent starts from the initial belief b_0. When the sequence of actions has infinite length, a discount factor $\gamma \in (0, 1)$ is specified, so that the total reward is finite and the problem is well defined.

The solution to a POMDP problem is a mapping from beliefs to the best actions, and is called an optimal policy. A policy π induces a value function $V_\pi(b)$, which specifies the expected total reward of executing policy π from belief b, and is computed as $V_\pi(b) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b, \pi]$. An optimal policy is the policy π^* whose value function $V_{\pi^*}(b)$ is the highest among all possible policies for any belief b.

A NOMDP is a class of POMDPs where observations are not available, i.e., $Z(s, a, na) = 1$ for any $s \in S$ and $a \in A$. As a consequence, the solution of an NOMDP is a nominal path, which maps time steps to the best actions.

The problem setups for both NOMDPs and POMDPs are geared towards motion planning problems, finding a strategy to move from one state to another. Both NOMDPs and POMDPs use the same representation for state and action spaces, and for motion uncertainty. The state space is a continuous metric space, denoted by $\mathbb{X}$, that is diffeomorphic to $\mathbb{R}^n$, where n is the dimensionality of $\mathbb{X}$. The action space is the same as the control space, denoted as $\mathbb{U}$, and typically has lower or equal dimension than $\mathbb{X}$. The motion uncertainty comes from actuation error, and is represented as a stochastic dynamical system of the form:

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x(t), \tilde{u}(t))dt, \tag{1}$$

where $x(t) \in \mathbb{X}$, $\tilde{u}(t) \in \mathbb{U}$ and Δt is a discrete time step. The control $\tilde{u}(t)$ is the one executed by the system, which does not always correspond to the input control $u(t)$ due to actuation error, i.e., there is an error vector w based on which:

$$\tilde{u}(t) = u(t) + w. \tag{2}$$

The vector w is additive noise sampled from a probability distribution, which can be any type of distribution.

Given that NOMDPs do not have observations, NOMDPs and POMDPs differ in the objective function. NOMDPs use an objective function that is commonly used in motion planning. In this paper, the defined NOMDP challenge involves finding a sequence of control inputs that minimizes the cost, while ensuring that the collision probability is lower than a given threshold and the probability of reaching the goal is higher than a given threshold. More precisely, suppose $p = (u_1, u_2, \ldots, u_m)$ is the sequence of input controls for the system. Each control input in the sequence is applied for a unit time step, subsequently starting from the initial belief b_0. This application will induce a trajectory $\pi_b^p = (b_0, b_1, b_2, \ldots, b_m)$ that arises when applying Eq. 1 to the states in the support set of belief b_0 and following the sequence p, given the noise model w. The duration of a trajectory π is denoted as T_π. If the trajectory π is induced by p with controls for m time steps, then $T_\pi = m \cdot \Delta t$. A belief state along

a trajectory π at time t is denoted as $\pi(t)$. The cost of the trajectory $\pi_{b_0}^p$ induced by p is $c(\pi_{b_0}^p) = \sum_{i=1}^{m} \int_{x \in \mathbb{X}} cost(x, u_i) \cdot b_{i-1}(x) dx$. The goal of the NOMDP solver is to find a control sequence p that generates a trajectory $\pi_{b_0}^p$ for which the probability of being in the goal region at time T_π is above the threshold: $\mathbb{P}(\pi(T_\pi) \in \mathbb{X}_G)) > \mathbb{P}_{goal}$, the probability for being in the free space is above the threshold: $\mathbb{P}_{free}(\pi) > \mathbb{P}_{valid}$, and which minimizes the cost $c(\pi)$.

The objective function of POMDPs uses the commonly applied definition (Sect. 2). The reward function, $R(s, a)$ for any pair of state s and action a, is a summation of collision cost and reward for being in the goal. The reason for this difference in objective function is that solving a POMDP with probability constraints is still a relatively open problem.

3 Distance Functions for Belief-Space Planning

Computing the optimal POMDP policy is computationally intractable [19]. In the past several years, however, methods which can compute a good approximation to the optimal policy fast have been proposed. Most of them rely on sampling, and depend on distance functions. They can be classified into several distinct approaches.

Point-based techniques [12, 23, 27, 28] use sampling to construct a small but representative set of beliefs, and find an approximately optimal policy by iteratively computing Bellman backups [23] on the sampled beliefs. The key is the sampling strategy, and some of the successful methods use distance functions to guide sampling. For example, PBVI [23] only keeps a newly sampled belief whenever the L1 distance between the new and existing belief is larger than a certain threshold, while GCS [13] uses EMD, and an alternative [8] uses KL divergence to perform similar rejection sampling. A fast offline point-based solver [12] also uses L1 distance for pruning. Point-based methods can handle any type of distributions but they have difficulties in solving problems in large continuous action spaces.

This work evaluates four distance functions for belief-space planning. Two of them, Kullback–Leibler (KL) and L1 divergence, are commonly used in belief-space planning. In general, these two functions do not consider the underlying state-space distance when computing distances between beliefs. This characteristic limits the effectiveness of these two distance functions to guide sampling and pruning in the belief space. To alleviate these problems, two alternatives are proposed here, Wasserstein (also known as Earth Mover's Distance (EMD)) and Hausdorff distance. Both of these functions compute distances based on an underlying state space distance. They have not been widely used in belief-space planning, but they are widely used in computer vision and optimal transport, which means efficient implementations for these distance computations abound. For the following, the beliefs are defined as distributions over a common state space, denoted as $\mathbb{X}$.

A. Wasserstein Distance/EMD: Intuitively, Wasserstein distance or EMD computes the distance between two distributions as the amount of work to move the probability mass of one distribution to another. More formally, EMD is defined as:

$$D_W(b, b') = \inf_f \left\{ \left| \int_{x \in \mathbb{X}} \int_{x' \in \mathbb{X}} d_{\mathbb{X}}(x, x') f(x, x') \partial x \partial x' \right| \right.$$
$$\left. b = \int_{x'} f(x, x') dx' \ , \ b'(x') = \int_x f(x, x') dx \right\}, \tag{3}$$

where $d_{\mathbb{X}}$ is the distance in the state space $\mathbb{X}$ and f is a joint density function.

EMD carries the Lipschitz continuity of a cost function in the state space to Lipschitz continuity of expected cost in the belief space. Formally, let $d_{\mathbb{X}}$ be the distance function on a separable state space $\mathbb{X}$ and $cost(x, u)$ be the cost of applying control $u \in \mathbb{U}$ at state $x \in \mathbb{X}$ for the duration of one unit time. Let beliefs b and b' be distributions over $\mathbb{X}$ and let the cost function w.r.t. a belief b be $cost(b, u) = \int_{x \in \mathbb{X}} cost(x, u) b(x) dx$.

Theorem For any control input $u \in \mathbb{U}$, if the cost function satisfies Lipschitz continuity in the state space, i.e., $|cost(x, u) - cost(x', u)| \leq C \cdot d_{\mathbb{X}}(x, x')$ for any $x \in \mathbb{X}$, then the cost function in the belief space with the EMD metric is also Lipschitz continuous: $|cost(b, u) - cost(b', u)| \leq C \cdot D_W(b, b')$.

Proof The proof is based on the well-known Kantorovich duality of the Wasserstein distance [5]. The Kantorovich distance is defined as

$$D_K(b, b') = \sup_{g \in Lip_1} \left(\int_{x \in \mathbb{X}} g(x) b(x) \, dx - \int_{x \in \mathbb{X}} g(x) b(x) \, dx \right),$$

where Lip_1 is the set of all 1-Lipschitz functions over $\mathbb{X}$. Now, by definition:

$$\left| cost(b, u) - cost(b', u) \right| = \left| \int_{x \in \mathbb{X}} cost(x, u) b(x) \, dx \ - \int_{x \in \mathbb{X}} cost(x, u) b'(x) \, dx \right|. \tag{4}$$

To satisfy the 1-Lipschitz requirement of the Kantorovich distance, we can use the scaled distance function in the state space $\mathbb{X}$, i.e., $d'_{\mathbb{X}} = C \cdot d_{\mathbb{X}}$. It is known that if we use $d'_{\mathbb{X}}$ as the state space distance, then the belief space distance $D'_W = C \cdot D_W = C \cdot D_K$. The last equality is due to the duality of the Kantorovich and Wasserstein distance. This means that by using the state space metric $d'_{\mathbb{X}}$, Eq. (4) can be bounded as: $\left| cost(b, u) - cost(b', u) \right| \leq C \cdot D_K(b, b')$, and hence $\left| cost(b, u) - cost(b', u) \right| \leq C \cdot D_W(b, b')$. □

From the literature [11], it is known that the value function in POMDPs is Lipschitz continuous when the metric in the belief space is EMD. The above theorem uses a similar proving strategy similar to the prior work [11] but generalizes the result to any cost function that is Lipschitz continuous in the state space.

B. **Hausdorff Distance**: Hausdorff distance is a popular metric in computer vision. This function computes distance between two sets based on a max-min operation. In terms of distances between beliefs, it is possible to define it with respect to the beliefs' support set. Slightly abusing the notation of the arguments, this function can be defined in the belief space as:

$$D_H(b, b') = \max\{d_H(b, b'), d_H(b', b)\}$$

$$\text{where } d_H(b, b') = \max_{x \in support(b)} \left\{ \min_{x' \in support(b')} \left\{ d_{\mathbb{X}}(x, x') \right\} \right\},$$

and $support(b) = \{x \in \mathbb{X} \mid b(x) > 0\}$, while $d_{\mathbb{X}}(x, x')$ is the state space distance.

Hausdorff is simpler to compute than EMD. Nevertheless, since Hausdorff measures distances between sets, rather than distributions, it ignores the probability values. This means that if two distributions have exactly the same support, even though the probabilities are significantly different, the two distributions will be considered to lie at the same point in the belief space. This problem is exactly the opposite of the problems faced by `L1` and `KL`-divergence, as described below.

C. `KL`**-Divergence**: A commonly used distance function in belief-space planning is the Kullback–Leibler (`KL`) divergence. It measures the difference in information content between two distributions. More formally, `KL` divergence is defined as:

$$D_{KL}(b, b') = \int_{x \in \mathbb{X}} b(x) \left(\ln b(x) - \ln b'(x) \right) dx. \tag{5}$$

In the general case, `KL`-divergence does not consider the underlying state space distance. But, for certain distributions, it does so to some extent. For instance, when applied to Gaussian beliefs, `KL`-divergence is partially based on the Euclidean distance of the mean. More completely, the `KL`-divergence between two multivariate Gaussian beliefs, denoted as $b_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $b_2 = \mathcal{N}(\mu_2, \Sigma_2)$, is

$$D_{KL}(b_1, b_2) = \frac{1}{2} \left((\mu_2 - \mu_1)^T \Sigma_1^{-1} (\mu_2 - \mu_1) + tr\left(\Sigma_2^{-1} \Sigma_1\right) + \ln \frac{|\Sigma_2|}{|\Sigma_1|} - K \right), \tag{6}$$

where K is the dimension of the underlying state space.

When applied to general beliefs with continuous state space, one usually starts by discretizing the state space $\mathbb{X}$ into uniform grid cells, and then computes the `KL`-divergence using Eq. (5) as if the beliefs are discrete distributions. This computation means that state-space distance is *only* considered up to the resolution of the grid cells, which is very limited.

Note that `KL` divergence is not symmetric and hence is not a true metric. One can symmetrize `KL` simply by adding the reverse distance or by computing distance to the mean of the two distributions (i.e., $\frac{D_{KL}(b, \frac{b+b'}{2}) + D_{KL}(b', \frac{b+b'}{2})}{2}$). The later strategy is called Jensen-Shannon divergence. In the accompanying comparative study, two

implementations are used: (i) the Jensen-Shannon and (ii) an approximation of the beliefs using Gaussian distributions per Eq. (6).

D. `L1` **Distance**: Another commonly used distance function in belief-space planning, and also the simplest to compute, is `L1`, which is defined as:

$$D_{L1}(b, b') = \int_{x \in \mathbb{X}} |b(x) - b'(x)| \, dx.$$

Most belief-space planners use `L1` distance for discrete distributions, that is $\mathbb{X}$ is discrete and `L1` is computed as a summation over $\mathbb{X}$ rather than an integration.

When the state space is continuous and `L1` distance is used, then the state space is discretized at a suitable resolution and the `L1` distance computation is applied as if the beliefs are distributions over the discretized state space. This discretization means that `L1` distance, similar to `KL` divergence, considers the underlying state-space distance *only* up to the resolution of the state space discretization, which is often very limited.

4 Algorithms for Comparative Study

To determine which distance function is most suitable for belief space planning given its complexity, we employ a sampling-based framework.

A. **Non-Observable Markov Decision Processes (`NOMDPs`)**: The framework follows tree sampling-based planners. It is based on a `BestNear` variant of `RRT` [17, 30] but is adapted to belief space planning. It has been formally shown - at least for the deterministic case - that this method can improve path quality over time even when there is no access to a steering function [15, 16]. Convergence to optimality requires Lipschitz continuity in the state and control spaces.

Algorithm 1: `SPARSE_BELIEF_TREE`($\mathbb{B}$, $\mathbb{U}$, b_0, N, T_{max}, δ_n, δ_s)

```
1 G = {V → {b_0}, E → 0};
2 for N iterations do
3     b_selected ← SelectNode (B,V,δ_n);
4     b_new ← Random_Prop (b_select, U, T_max);
5     if IsNodeLocallyBest (b_new, S, δ_s) then
6         V ← V ∪ {b_new};
7         E ← E ∪ {b_select → b_new};
8         Prune_Dominated_Nodes(b_new, V, E, δ_s);
```

An outline of the algorithmic framework is shown in Algorithm 1. As input, the planner receives the belief space $\mathbb{B}$, control space $\mathbb{U}$, initial belief b_0 and number

of iterations N. In addition, Algorithm 1 receives a maximum propagation duration T_{max} and two radius parameters δ_n and δ_s, which are explained below. The selection process (Line 3) is summarized in Algorithm 2. A random δ-belief distribution b_{rand} is first sampled in the belief space $\mathbb{B}$ and the set of belief distributions B_{near} within a distance threshold δ_n is computed. If no belief is found within this threshold, then the closest distribution is returned, similar to the basic `RRT` approach. If there are beliefs in the set $D_{\mathbb{B}}$, then the one with the best cost, e.g., trajectory duration, is returned.

Algorithm 2: `SelectNode`($\mathbb{B}$, V, δ_n)

1 $b_{rand} \leftarrow$ `Sample_Belief`($\mathbb{B}$);
2 $B_{near} \leftarrow$`Near`(V, b_{rand}, δ_n);
3 **If** $B_{near} = \emptyset$ **return** `Nearest`($\mathbb{V}$, b_{rand});
4 **Else return** $\arg\min_{b \in B_{near}}$ `cost`(b);

Line 4 of Algorithm 1 is the propagation primitive used to add new belief states to the tree. The subroutine is detailed in Algorithm 3. First, a time duration is uniformly sampled up to a maximum time T_{max}. The sampled time must be a constant multiple of the minimum Δt in order to satisfy the requirement for piece-wise constant control inputs, which are sampled after the time duration. Given these control inputs, the belief distribution can be updated through the transition model.

Algorithm 3: `Random_Prop`(b_{prop}, $\mathbb{U}$, T_{max})

1 $t \leftarrow$ Sample$(0, T_{max})$; $\Upsilon \leftarrow$ Sample$(\mathbb{U}, t)$;
2 **return** $b_{new} \leftarrow \int_0^t \mathbb{T}(b(t), \Upsilon(t)) b_{prop}\, dt$;

To perform pruning, the set of nodes V in the tree data structure $G(V, E)$ of algorithm Algorithm 1 are split into two subsets V_{active} and $V_{inactive}$. Nodes in V_{active} have the best cost from the start in a neighborhood of radius δ_s around them. These nodes are considered for propagation by the algorithm. Nodes that are dominated by others in terms of path cost in their local neighborhood, are:

- Pruned if they are leaves or have no children in V_{active}.
- Or added to the set $V_{inactive}$ if they do have children in V_{active}. Nodes in $V_{inactive}$ are not selected for propagation.

Algorithm 4 details a simple operation to determine how to prune existing nodes in the tree. It is called only if the new belief distribution b_{new} has the best cost in its local δ_s neighborhood. Then, the set of existing nodes that are dominated in terms of path cost are set to be inactive. If these nodes are also leaves of the tree, they are removed from the tree. This process can continue up the tree if the parents were also inactive. It helps to reduce the size of the stored tree and promotes the selection of nodes with good path quality.

Algorithm 4: `Pruning`(b_{new}, G, δ_s)

```
1 B_dominated ← FindDominated(G, b_new, δ_s);
2 for b ∈ B_dominated do
3     b.set_inactive();
4     while IsLeaf (b) and b is inactive do
5         x_parent ←Parent(b);
6         E ← E \ {b_parent → b};
7         V ← V \ {b};
8         b ← b_parent;
```

It is apparent that throughout the operation of this sampling-based framework for belief space planning, there is heavy use of distance calls and a significant dependence on the choice of the distance function.

B. Partially Observable Markov Decision Processes (`POMDPs`): The framework follows a point-based approach, similar to Monte Carlo Value Iteration (`MCVI`)[2], an extension of `SARSOP` [12]. The later is considered the fastest offline and general `POMDP` solver for problems with continuous state space. The `MCVI` method is slightly modified to use the distance function for pruning sampled beliefs. Algorithm 5 details the algorithm employed for `POMDPs`, which incorporates `SARSOP` as its sampling strategy (Line 6). The function $Nearest(b)$ in Line 7 and 8 returns the belief in the tree $\mathscr{T}$ that is nearest to belief b. The only modification made to `MCVI` is the addition of the condition in Line 7, which rejects a newly sampled belief whenever its nearest neighbor in $\mathscr{T}$ is within a given threshold.

Algorithm 5: `ModifiedMCVI`(b_0)

```
1  Initialize belief tree 𝒯 by setting b_0 as the root of 𝒯 ;
2  Initialize policy graph Γ with an empty graph ;
3  Initialize the upper bound V̄ of the value function to be ∞ ;
4  Initialize the lower bound V̲ of the value function to be -∞ ;
5  while |V̄(b_0) − V̲(b_0)| > ε do
6      Sample new belief b ;
7      if distance(b, Nearest(b)) < δ_th then
8          b ← Nearest(b) ;
9      else
10         Add b to 𝒯 ;
11     MCVI Backup(Γ, b) ;
12     V̄ = UpdateUpperBound(b) ;
13     V̲ = UpdateLowerBound(b) ;
```

C. Algorithmic Details to Improve Speed: The implementation of the distance functions was optimized to reduce computation time. `KL` and `L1` distances were

implemented through the use of binning. The full state space grid is not required, only the nonzero entry bins. This allows dealing with high-dimensional problems, saves space, and reduces computation time. The Hausdorff and EMD functions do not require binning but become much more efficient using bins. The bin width is chosen so that several individual bins can be contained within the pruning radius δ_s when solving NOMDPs. This discretization introduces an approximation error.

To further speed up the EMD computation, an approximation is employed to occasionally replace the expensive call to the standard method. The motivation stems from the fact that if two distributions are too far apart, their EMD distance is close to the distance between their centroids. So, if two distributions overlap according to their diameters and their discretization, then the standard call to the EMD computation is performed. Otherwise, the distance between the two centroids is used, which is a fast operation.

In the following experiments KL-Gaussian represents the approximation of a set of particles as a Gaussian distribution and performs distances using the closed form expression from Eq. 6. This distance function does not use binning as the Gaussian parametrization provides an efficient representation.

5 Experimental Evaluation

5.1 *Non-observable Markov Decision Processes (NOMDPs)*

All distances are evaluated in the scenarios shown in Fig. 2. All scenarios produce non-Gaussian belief distributions due to the nonlinear dynamics. The objective is to reach a goal region in state-space with at least 90% probability. Valid trajectories have a collision probability of less than 20%.

2D Rigid Body. This introductory example is a 2D rigid body moving among two narrow corridors. Due to errors in actuation and requirement for collision avoidance, the robot can only move through the lower corridor. The state space is 2D (x, y) and the control space is also 2D (v, θ), where $v \in [0, 10]$ and $\theta \in [-\pi, \pi]$. The dynamics

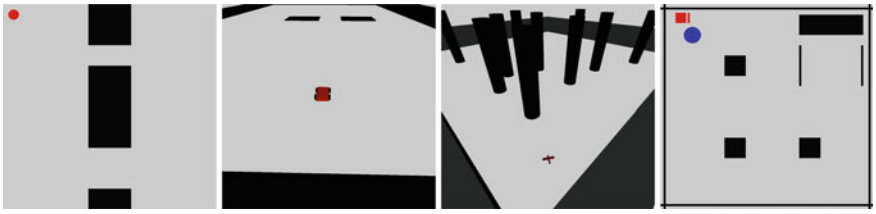


Fig. 2 The considered scenarios, which are better viewed in color. The 2D rigid body (*left*) must move from the *left* to the *right* side. The car (*left middle*) must drive through one of 3 corridors. The fixed-wing airplane (*middle right*), must move to the opposite corner while avoiding cylinders. The manipulator (*right*) must push the round object into the storage location at the *top right*

follow this model:

$$\dot{x} = \tilde{v}\cos(\tilde{\theta}), \quad \dot{y} = \tilde{v}\sin(\tilde{\theta}),$$

where $\tilde{v} = v + \mathcal{N}(0, 1)$ and $\tilde{\theta} = \theta + \mathcal{N}(0, 0.3)$. Numerical integration is performed using Euler integration.

2^{nd}**-order Car**. A four-wheeled vehicle with dynamics needs to reach a goal region, while ensuring low collision probability. The state space is 5D $(x, y, \theta, v, \omega)$, the control space is 2D $(a, \dot{\omega}) \in ([-1, 1], [-2, 0.2])$, actuation error is $(\mathcal{N}(0, 0.05), \mathcal{N}(0, 0.002))$, and the dynamics are:

$$\dot{x} = v\cos(\theta)\cos(\omega), \quad \dot{y} = v\sin(\theta)\cos(\omega),$$
$$\dot{\theta} = v\sin(\omega), \qquad \dot{v} = \tilde{a}.$$

Numerical integration is performed using Runge-Kutta order four (RK4). The environment is more complex than before since there are multiple feasible paths through each of the 3 corridors.

Fixed-wing airplane. An airplane flying among multiple cylinders. The state space is 9D $(x, y, z, v, \alpha, \beta, \theta, \omega, \tau)$, the control space is 3D $(\tau_{des} \in [4, 8], \alpha_{des} \in [-1.4, 1.4], \beta_{des} \in [-1, 1])$ and the dynamics are (from [21]):

$$\dot{x} = v\cos(\omega)\cos(\theta), \quad \dot{y} = v\cos(\omega)\sin(\theta)), \quad \dot{z} = v\sin(\omega),$$
$$\dot{v} = \tau * \cos(\beta) - C_d k v^2 - g\sin(\omega), \quad \dot{\omega} = \cos(\alpha)(\frac{\tau\sin(\beta)}{v} + C_l k v) - g\frac{\cos(\omega)}{v},$$
$$\dot{\theta} = v\frac{\sin(\alpha)}{\cos(\omega)}(\frac{\tau\sin(\beta)}{v} + C_l k v), \quad \dot{\tau} = \widetilde{\tau_{des}} - \tau \quad \dot{\alpha} = \widetilde{\alpha_{des}} - \alpha, \quad \dot{\beta} = \widetilde{\beta_{des}} - \beta,$$

where $\widetilde{\tau_{des}} = \tau_{des} + \mathcal{N}(0, 0.03)$, $\widetilde{\alpha_{des}} = \tau_{des} + \mathcal{N}(0, 0.01)$, and $\widetilde{\beta_{des}} = \beta_{des} + \mathcal{N}(0, 0.01)$. Numerical integration is performed using RK4. This problem has a state space that is generally larger than most planners in belief space can handle computationally. Leveraging sampling-based techniques with proper distance functions makes planning for the airplane model possible.

Non-prehensile manipulator. The task is to push an object to the goal. The state space is 5D $(x_{man}, y_{man}, x_{obj}, y_{obj}, \theta_{manip})$ and the control space is 2D (v, θ), where $v \in [0, 10]$ and $\theta \in [-\pi, \pi]$. The dynamics are:

$$\dot{x}_{man} = \tilde{v}\cos(\tilde{\theta}), \qquad \dot{y}_{man} = \tilde{v}\sin(\tilde{\theta}),$$

where $\tilde{v} = v + \mathcal{U}(-1, 1)$ and $\tilde{\theta} = \theta + \mathcal{U}(-0.3, 0.3)$. Numerical integration is performed using RK4. The object cannot be moved unless the manipulator moves in the direction of the object and pushes it, which implies that there is contact between the manipulator and the object. Once pushed, the object moves as if it is attached to the manipulator. Notice that the noise model used in this setup is a uniform distribution, meaning that the resulting belief distributions are clearly non-Gaussian.

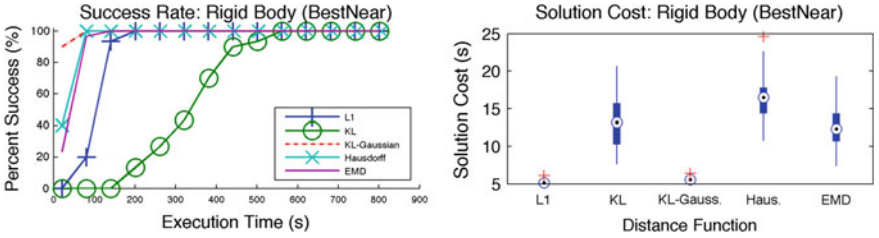


Fig. 3 Results for the 2D rigid body - better viewed in color

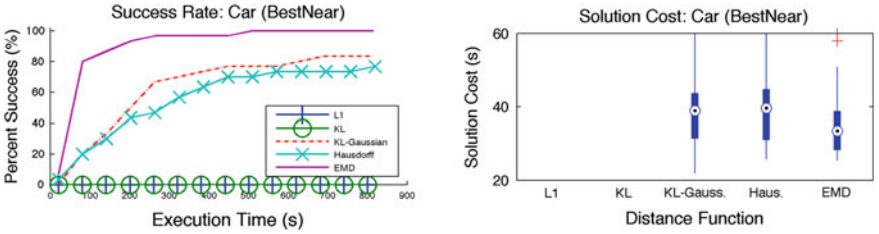


Fig. 4 Results for the car. L1 and KL failed to produce solutions within the time constraint

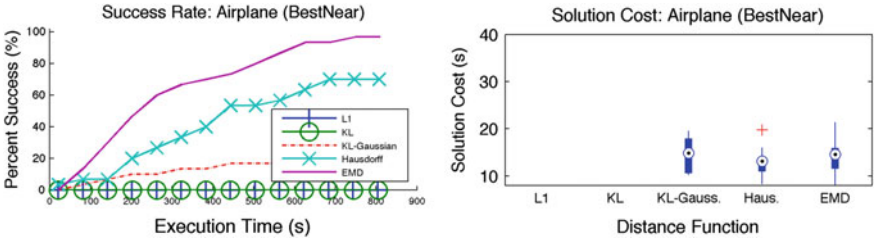


Fig. 5 Results for the airplane. L1 and KL failed to produce solutions within the time constraint

Experimental Setup and Results: Each combination of distance function and algorithm is evaluated in each of the four scenarios described in the previous subsection. The key criterion is the success rate for finding solutions in the allotted time. The distance thresholds in the algorithms for each metric are selected so that a similar number of nodes is kept among all alternatives. This allows for a fair comparison on the effect of a metric to the quality of sample placement. The experiments were executed on Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz machines. Each experiment is repeated 30 times with different random seeds. The results are averaged over these runs and are presented in Figs. 3, 4, 5 and 6.

In most scenarios, belief metrics that do consider the underlying state-space distance, such as `EMD`, Hausdorff, and `KL`-Gaussian, perform significantly better than those that do not. `KL` and `L1` metrics consider the state space distance up to the resolution of the state space discretization (as described in Sect. 4). Such consideration is sufficient when the state space is small (as in Fig. 3). As the size of the state

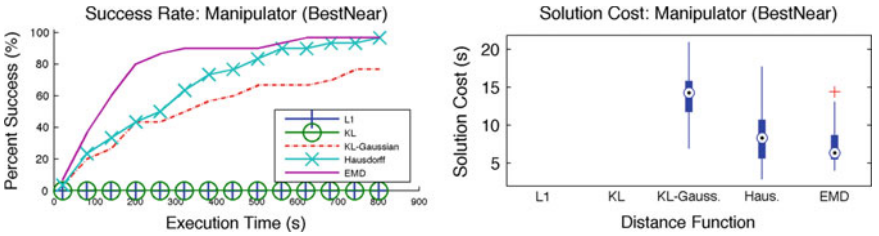


Fig. 6 Manipulation results. L1 and KL failed to produce solutions within the time constraint

space increases, this is no longer sufficient. These results corroborated the hypothesis regarding the importance of taking into account the underlying state space distance in computing distance between beliefs.

`EMD` performs substantially better than the alternatives since it considers both state space distance and the distributions. The Hausdorff distance performs better than `L1` and `KL` because it still considers the underlying state-space distance. But Hausdorff does not consider the distribution, and therefore is outperformed by `EMD`. `KL`-Gaussian performs better than `L1` and `KL` because it considers the mean and variance of the corresponding Gaussian and in this way considers the underlying state space distance. Nevertheless, the usefulness of this distance function depends on how well a Gaussian distribution approximates the actual distribution.

The path costs achieved by the different metrics for successful runs are comparable. Therefore, the success rate is the key criterion for evaluating the effect of the different metrics to the performance of belief-space planners.

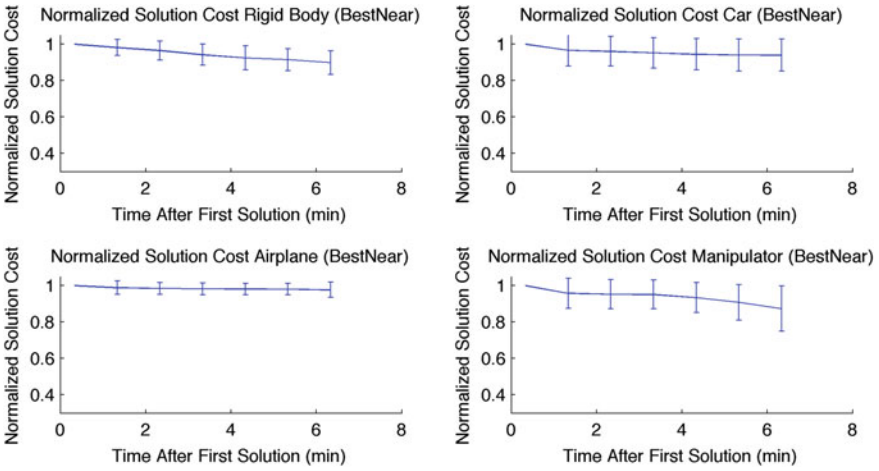


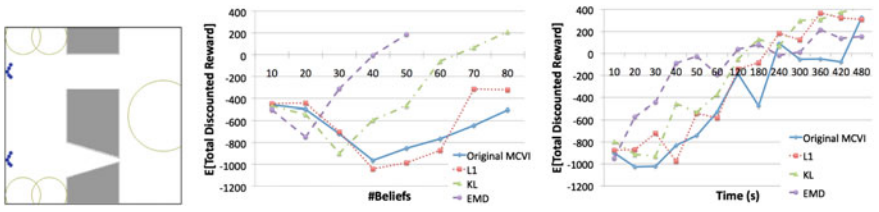
Fig. 7 The normalized costs over time for `EMD` show the benefit of providing improving solutions. Each trial's initial solution is normalized to 1. The cost over time is shown with one st. dev

Results on Cost Reduction Over Time: Figure 7 normalizes to one the first solution generated for each individual run of the algorithm using EMD. Then, all subsequent improvements to the path cost are plotted relative to the initial cost. The figure averages all runs, all of which show improvement over time. The improvement is most prominent for the manipulator experiments. This comparison is performed only for EMD as it provides the best performance. Path cost can also improve over time using Hausdorff and KL-Gaussian but there are fewer data points to extract useful conclusions given the reduced success ratio of these metrics.

5.2 *Partially Observable Markov Decision Processes (POMDPs)*

The different distance functions are tested on a 2D navigation problem (Fig. 8a). The robot is a point robot, and may start from any position marked by the blue region. Its task is to reach the goal region (the large circular region on the right) as fast as possible while avoiding collision with obstacles (the grey polygons) as much as possible. The robot's motion is discretized into 5 actions, which is moving a unit distance in the direction of N, NE, E, SE, and S. Its motion has error, which is represented as a bounded uniform distribution. The robot can only localize itself up to a certain accuracy inside the small circular regions on the left of the obstacles. The problem is modeled as a POMDP with continuous state space, but with discrete action and observation spaces. The reward function of the POMDP model is a sum of the goal reward, collision penalty, and moving cost.

Experimental Setup: The original and modified MCVI method (Algorithm 5) is evaluated separately for the L1, KL, and EMD metrics. To set the required parameters, such as distance threshold, short preliminary runs with various parameters were executed. The best parameters for each method were retained. Each method was then executed with the appropriate parameters to generate 15 different sets of policies. To generate each set of policies, the method is ran until the difference between the upper and lower bound of the initial belief is less than a given threshold, so that at the end of the runs, the quality of the policies generated by the different methods are similar. Throughout each run, the method outputs the intermediate policies at every



Scenario Results

Fig. 8 Comparative study on a POMDP problem

time interval and at every time the policy graph reaches a certain size. Each policy is then evaluated through 1, 000 simulation runs. The expected total discounted reward of a method is computed as the average total discounted reward over all simulation runs of all the policies generated by the same method within the same interval.

Results: The results are presented in Fig. 8b–c. They indicate that EMD significantly improves placement of sampled beliefs. Nevertheless, the benefit of using EMD degrades over time. The reason is that a naive implementation of distance computation is used, checking the distance between a new sampled belief and all beliefs that are already in the belief tree. As time increases, the size of the belief tree increases, and so is the time taken for this computation. This step affects the EMD computation much more than other metrics because each EMD evaluation is more expensive. Nevertheless, there has been a lot of work on speeding up EMD computation [22, 26], which can help to alleviate this problem.

It would be interesting to see how the POMDP evaluation with EMD performs on problems with similar scale as the NOMDP scenarios. Nevertheless, existing POMDP solvers cannot solve problems with action spaces and planning horizon as large and as long as the examples used in the case of NOMDPs (i.e., beyond the 2D rigid body scenario). Nevertheless, as shown in the case of the NOMDP results, the benefit of EMD becomes more visible as the problem becomes more complex. This is likely to be true for POMDPs as well. The cost of reward computation in the POMDP test case is negligible. In more complex motion planning problems, the reward computation often includes expensive collision checks. In such problems, the ability to solve problems with smaller number of sampled beliefs, which reduces the number of backup operations (and hence reward computations), would be more beneficial. Therefore, the expectation is that EMD would show additional benefits when the POMDP problems represent more complex planning problems.

6 Discussion and Conclusion

This work demonstrates that using the Wasserstein distance in belief space planning provides significant improvements over commonly used alternatives, such as Kullback–Leibler divergence and L1 distance. This is especially apparent when planning for higher-dimensional systems. By considering an appropriate metric in belief space, it is possible to gain benefits from recent advances in sampling-based planning, which allow the computation of trajectories of increasing path quality.

With the rise of new methods for belief space planning, it is time to take a step back to understand critical components that make belief space planners perform well. This paper is a preliminary attempt to provide such an insight.

References

1. Bai, H., Hsu, D., Kochenderfer, M.J., Lee, W.S.: Unmanned aircraft collision avoidance using continuous-state POMDPs. Robot. Sci. Syst. **1**, 1–8 (2012)
2. Bai, H.Y., Hsu, D., Lee, W.S., Ngo, V.A.: Monte Carlo value iteration for continuous-state POMDPs. In: Hsu, D., et al. (eds.) WAFR. Springer, Heidelberg (2010)
3. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: ICRA (2011)
4. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion. The MIT Press, Cambridge (2005)
5. Dudley, R.M.: Real Analysis and Probability. Cambridge University Press, Cambridge (2002)
6. Horowitz, M., Burdick, J.: Interactive non-prehensile manipulation for grasping via POMDPs. In: Proceedings of the IEEE International Conference on Robotics and Automation (2013)
7. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. IJRR **30**(7), 846–894 (2011)
8. Kneebone, M., Dearden, R.: Navigation planning in probabilistic roadmaps with uncertainty. In: Proceedings of the International Conference on Automated Planning and Scheduling (2009)
9. Koval, M.C., Pollard, N.S., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In: RSS (2014a)
10. Koval, M., Pollard, N., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In: RSS, Berkeley, USA, July 2014b
11. Kurniawati, H., Patrikalakis. N.M.: Point-based policy transformation: adapting policy to changing POMDP models. In: WAFR (2012)
12. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: RSS (2008)
13. Kurniawati, H., Bandyopadhyay, T., Patrikalakis, N.M.: Global motion planning under uncertain motion, sensing, and environment map. Auton. Robots Spec. Issue RSS **30**(3), 2012 (2011)
14. Kurniawati, H., Du, Y., Hsu, D., Lee, W.S.: Motion planning under uncertainty for robotic tasks with long time horizons. IJRR **30**(3), 308–323 (2011)
15. Li, Y., Littlefield, Z., Bekris, K.E.: Sparse methods for efficient asymptotically optimal kinodynamic planning. In: WAFR (2014)
16. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. In: IJRR (2015), accepted to appear
17. Littlefield, Z., Li, Y., Bekris, K.E.: Efficient sampling-based motion planning with asymptotic near-optimality guarantees for systems with dynamics. In: IROS, Tokyo Big Sight, Japan (2013)
18. Melchior, N., Simmons, R.: Particle RRT for path planning with uncertainty. In: ICRA (2007)
19. Papadimitriou, C., Tsitsiklis, J.N.: The complexity of Markov decision processes. JMOR **12**(3), 441–450 (1987)
20. Papadopoulos, G., Kurniawati, H., Patrikalakis, N.M.: Analysis of asymptotically optimal sampling-based motion planning algorithms for Lipschitz continuous dynamical systems, 12 May 2014. http://arxiv.org/abs/1405.2872
21. Paranjape, A.A., Meier, K.C., Shi, X., Chung, S.-J., Hutchinson, S.: Motion primitives and 3-D path planning for fast flight through a forest. In: IROS (2013)
22. Pele, O., Werman, M.: Fast and robust earth mover's distances. In: ICCV (2009)
23. Pineau, J., Gordon, G., Thrun, S.: Pointspsbased value iteration: an anytime algorithm for POMDPs. In: IJCAI (2003)
24. Platt, R., Kaelbling, L., Lozano-Perez, T., Tedrake, R.: Non-gaussian belief space planning: correctness and complexity. In: ICRA (2012)
25. Prentice, S., Roy, N.: The belief roadmap: efficient planning in belief space by factoring the covariance. IJRR **28**(11–12), 1448–1465 (2009)
26. Shirdhonkar, S., Jacobs, D.: Approximate earth mover's distance in linear time. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
27. Smith, T., Simmons, R.: Point-based POMDP algorithms: improved analysis and implementation. In: Proceedings of the Uncertainty in Artificial Intelligence (2005)

28. Spaan, M.T.J., Vlassis, N.: Perseus: randomized point-based value iteration for POMDPs. J. Artif. Intell. Res. **24**, 195–220 (2005)
29. Thrun, S.: Monte-Carlo POMDPs. In: NIPS (2000)
30. Urmson, C., Simmons, R.: Approaches for heuristically biasing RRT growth. In: IROS, pp. 1178–1183 (2003)
31. van den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. In: RSS (2010)
32. van den Berg, J., Patil, S., Aterovitz, R., Abbeel, P., Goldberg, K.: Planning, sensing, and control of steerable needles. In: Workshop on the Algorithmic Foundation of Robotics (2010)