# Learning Sampling Distributions for Robot Motion Planning

Brian Ichter*,1, James Harrison*,2, Marco Pavone1

***Abstract*— A defining feature of sampling-based motion planning is the reliance on an implicit representation of the state space, which is enabled by a set of probing samples. Traditionally, these samples are drawn either probabilistically or deterministically to *uniformly* cover the state space. Yet, the motion of many robotic systems is often restricted to "small" regions of the state space, due to e.g. differential constraints or collision-avoidance constraints. To accelerate the planning process, it is thus desirable to devise *non-uniform* sampling strategies that favor sampling in those regions where an optimal solution might lie. This paper proposes a methodology for non-uniform sampling, whereby a sampling distribution is learnt from demonstrations, and then used to bias sampling. The sampling distribution is computed through a conditional variational autoencoder, allowing sample generation from the latent space conditioned on the specific planning problem. This methodology is general, can be used in combination with any sampling-based planner, and can effectively exploit the underlying structure of a planning problem while maintaining the theoretical guarantees of sampling-based approaches. Specifically, on several planning problems, the proposed methodology is shown to effectively learn representations for the relevant regions of the state space, resulting in an order of magnitude improvement in terms of success rate and convergence to the optimal cost.**

## I. Introduction

Sampling-based motion planning (SBMP) has emerged as a successful algorithmic paradigm for solving high-dimensional, complex, and dynamically-constrained motion planning problems. A defining feature of SBMP is the reliance on an implicit representation of the state space, achieved through sampling the feasible space and probing local connections through a black-box collision checking module. Traditionally, these samples are drawn either probabilistically or deterministically to *uniformly* cover the state space. Such a sampling approach allows arbitrarily accurate representations (in the limit of the number of samples approaching infinity), and thus allows theoretical guarantees on completeness and asymptotic optimality. However, many robotic systems only operate in small subsets of the state space, which may be very complex and only implicitly defined. This might be due to the environment (e.g., initial and goal conditions, narrow passageways), the system's dynamics (e.g., a bipedal walking robot's preference towards stable, upright conditions), or implicit constraints (e.g., loop closures, multirobot systems remaining out of collision). The performance of SBMP is thus tied to the placement of samples in these promising regions, a result uniform sampling is only able to achieve through sheer exhaustion. Therein lies a fundamental challenge of SBMP algorithms: their implicit representation of the state space, while beneficial due to its generality and minimal set of assumptions, is limiting towards their ability to use known information about the workspace or robotic system, or to leverage knowledge gained from previous planning problems to accelerate solutions.

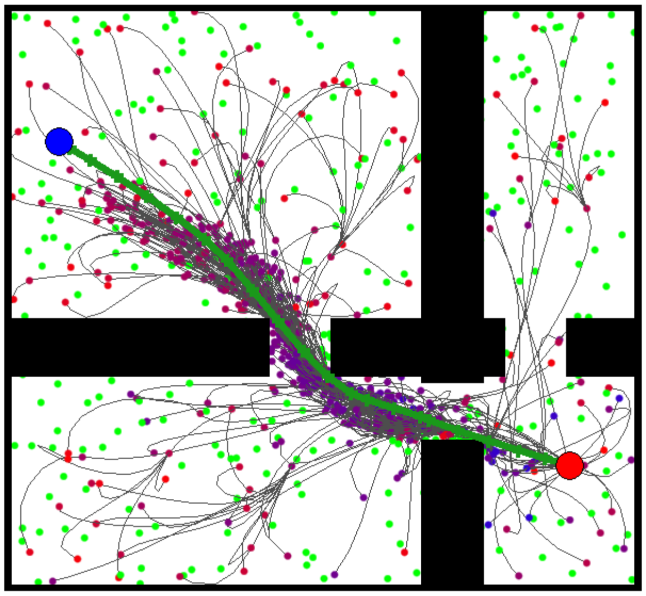


Fig. 1: A fast marching tree (FMT*) generated with learned samples for a double integrator, conditioned on the initial state, goal state, and workspace obstacles. Note the significantly higher density of samples along the solution plan.

In this work we approach this challenge through biasing the sampling of the state space towards these promising regions via *learned* sample distributions (see Fig. 1). At the core of this methodology is a conditional variational autoencoder (CVAE) [1], which is capable of learning complex manifolds and the regions around them, and is trained from demonstrations of successful motion plans and previous robot experience, rather than expert-crafted heuristics. The latent space of the CVAE can then be sampled and projected into a representation of the operational regions of the robotic system, conditioned on information specific to a given planning problem, for example, initial state, goal region, and obstacles (as defined in the workspace). In this way one can balance the theoretical and exploration benefits of sampling-based motion planning with the generality, extensibility, and practical performance associated with hyperparametric learning approaches. Remarkably, this methodology is extensible to any system and available problem-specific information.

*Related Work*: Several previous works have presented non-uniform sampling strategies for SBMP, often resulting in significant performance gains [2]. Recent work by Gammell et al. [3] adaptively places batches of samples based on solution information gained while planning [3]. Other works

* Brian Ichter and James Harrison contributed equally to this work.

1Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305. {ichter, pavone}@stanford.edu

2Department of Mechanical Engineering, Stanford University, Stanford, CA 94305. jh2@stanford.edu

leverage models of the workspace to bias samples via decomposition techniques [4], [5] or via approximation of the medial axis [6]. While these sampling heuristics are effective in the problems for which they were designed, it is often unclear how they perform on environments that are outside of their expected operating conditions [7]. This is particularly the case for planning problems that must sample the full state space (e.g., that include velocity). These methods further require significant expert intuition, while our method is able to compute efficient non-uniform distributions from only demonstration.

Several approaches to adaptive sampling and biased sampling have used online learning to improve generalization. Burns and Brock [8] aim to optimally sample the configuration space based on maximizing a given utility function for each new sample, related to the quality of configuration space representation. Zucker et al. [9] use a reinforcement learning approach to learn where to sample in a discretized workspace. Berenson et al. [10] cache paths, and use a learned heuristic to store new paths as well as recall and repair old paths. Coleman et al. [11] use a related experience-based approach to store experiences in a graph rather than as individual paths. Li and Kostas [12] learn cost-to-go metrics for kinodynamic systems, which often do not have an inexpensive-to-compute metric, especially for nonholonomic systems. Baldwin and Newman [13] learn distributions that leverage semantic information. Ye and Alterovitz [14] present a learning-based approach that leverages human demonstrations and SBMP to generate plans. Each of these approaches either fails to generalize to arbitrary planning problems or cannot leverage both the full state (e.g., velocity information) and external (e.g., obstacle information) factors. In this work we propose the use of recent advances in representation learning due to their ability to represent general, high-dimensional, complex conditional distributions and thus include all available state and problem information.

Representation learning (also known as feature learning or manifold learning, and generally considered a subset of deep learning or probabilistic graphical models [15]) has seen increasing use in robotics in the last decade [16]. This field primarily aims to extract features from unstructured data, to either achieve a lower dimensional representation (often referred to as encoding) or learn features for supervised learning or reinforcement learning. In the context of motion planning, Havoutis and Ramamoorthy [17] present an algorithm to identify manifolds that were used to focus sampling, based on Locally Smooth Manifold Learning [18]. This approach relies on a relatively expensive gradient descent approach and it is unclear how it may be combined with problem-specific information. Chen et al. [19] use a time-dependent variational autoencoder to learn dynamic motion primitives for humanoids, which, though it is not addressed, could be used toward sampling for the motion planning problem. In this work, we aim to develop a general methodology to incorporate environmental knowledge (which changes over the operation of the robot) with learned representations of the robotic system in question, to improve the quality of samples used to generate motion plans.

*Statement of Contributions:* The contributions of this paper are threefold. First, we present a methodology to learn sampling distributions for sampling-based motion planning. Such learned sampling distributions can be used to focus sampling on those regions where an optimal solution may lie, as dictated by both system-specific and problem-specific constraints. This methodology is general to arbitrary systems and problems, scales well to high dimensions, and maintains the typical theoretical results of SBMP (chiefly, completeness and asymptotic optimality). Second, we demonstrate the learned sampling distribution methodology on a wide variety of problems, from low-dimensional to high-dimensional, and from a geometric setting to problems with complex dynamics. Third, we compare our methodology to uniform sampling, finding approximately an order of magnitude improvement in success rates and path costs, as well as state of the art approaches for sample biasing. Our findings show analogous benefits to those seen recently in the computer vision community, where feature learning-based approaches have resulted in substantially improved performance over human intuition and handcrafted features.

*Organization:* The remainder of this document is organized as follows. Section II reviews the problem addressed herein. Section III outlines the learning sampling distribution methodology. Section IV demonstrates the performance of the method through numerical experiments. Section V summarizes the results and provides directions for future work.

## II. Problem Statement

The goal of this work is to generate sample distributions to aid sampling-based motion planning algorithms in solving the optimal motion planning problem. Informally, solving this problem entails finding the shortest free path from an initial state to a goal region, if one exists. The simplest version of the problem, referred to herein as the geometric motion planning problem, is defined as follows. Let $\mathcal{X} = [0,1]^d$ be the state space, with $d \in \mathbb{N}, d \geq 2$. Let $\mathcal{X}_{\text{obs}}$ denote the obstacle space, $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ the free state space, $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ the initial condition, and $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ the goal region. A path is defined by a continuous function, $\sigma : [0,1] \to \mathbb{R}^d$. We refer to a path as *collision-free* if $\sigma(\tau) \in \mathcal{X}_{\text{free}}$ for all $\tau \in [0,1]$, and *feasible* if it is collision-free, $\sigma(0) = x_{\text{init}}$, and $\sigma(1) \in \mathcal{X}_{\text{goal}}$. We thus wish to solve,

**Problem 1** (Optimal motion planning). *Given a motion planning problem $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ and a cost measure $c$, find a feasible path $\sigma^*$ such that $c(\sigma^*) = \min\{c(\sigma) : \sigma$ is feasible$\}$. If no such path exists, report failure.*

Generally, there exist formulations of this problem for systems with kinematic, differential, or more complex constraints, for which we refer the reader to [20], [21]. The general form of the motion planning problem is known to be PSPACE-complete [21], and thus one often turns to approximate methods to solve the problem efficiently. Sampling-based motion planning has achieved particular success in solving complex, high-dimensional planning problems. These algorithms (e.g., PRM$^*$, RRT$^*$ [22], FMT$^*$ [23]) approach the complexity of the motion planning problem by only implicitly representing the state space with a set of probing samples in $\mathcal{X}_{\text{free}}$ and making local, free connections to neighbor samples (sampled states within an $r_n > 0$ cost radius, where $n$ is the number of samples). As more samples are added, the implicit representation is able to model the true state space arbitrarily well, allowing theoretical guarantees of both probabilistic completeness (the probability that a solution will be found, if one exists, converges to one as the number of samples approaches infinity) and asymptotic optimality (the cost of the found path converges almost surely to the optimum) [22].

At the core of these algorithms is a sampling strategy from which the implicit representation is constructed. These samples are drawn from a sample source (random or deterministic) and then distributed over the state space according to a probability distribution [2]. Sampling sources, while traditionally independently and identically distributed random points, can be drawn through low-dispersion and deterministic sampling strategies allowing for improved theoretical guarantees and practical performance [24]. However, compared to the sampling distribution, the impact of the sample source is limited [2]. This work focuses primarily on computing sampling distributions to allocate samples to regions more likely to be in an optimal motion plan.

## III. Learning-Based Sample Distributions

The goal of this work is to develop a methodology capable of identifying subspaces of the state space containing optimal trajectories with high probability, and generating samples from this subspace to improve the performance of sampling-based motion planning (SBMP) algorithms. These regions may be arbitrary and potentially complex, defined by internal or external factors. Specifically, we refer to intrinsic properties of the robotic system, independent of the individual planning problem (e.g., the system dynamics) as internal factors, and we refer to properties specific to the planning problem itself (e.g., the obstacles, the environment, the initial state, and the goal state) as external factors. At the core of our methodology is a Conditional Variational Autoencoder (CVAE), as it is expressive enough to represent very complex, high-dimensional regions and general enough to admit arbitrary problem inputs. The CVAE is an extension of the standard variational autoencoder, which is a class of generative models that has seen widespread application in recent years, particularly in the image and video processing communities [25]. This extension allows conditional data generation by sampling from the latent space of the model [1]; in the motion planning context, the conditioning variables represent external factors. Lastly, these samples are used, along with uniform samples that ensure state space coverage, as the sampling distribution for SBMP algorithms. The method thus leverages previous robotic experience (motion plans and demonstrations) to inform planning algorithms. This combination of learning and SBMP allows both the generality of learning and the exhaustive exploration ability and theoretical guarantees of SBMP.

We now examine the methodology in detail, following along with the outline below. It begins with an offline phase which trains the CVAE, to be later sampled from. Line 1 initializes this phase with the required demonstration data. This data (states and any additional planning problem

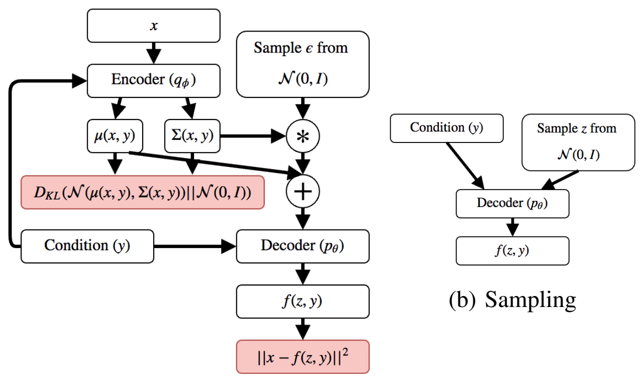
**Learning Sample Distribution Methodology Outline**

```
Offline:
  1 Input: Data (successful motion plans, robot in action,
    human demonstration, etc.)
  2 Construct conditioning variables $y$
  3 Train CVAE, as in Fig. 2a
Online:
  4 Input: Motion planning problem $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$,
    learned sample fraction $\lambda$
  5 Construct conditioning variable $y$
  6 Generate $\lambda N$ free samples from the CVAE latent space
    conditioned on $y$, as in Fig. 2b
  7 Generate $(1-\lambda)N$ free samples from an auxiliary
    (uniform) sampler
  8 Run sampling-based planner (e.g., PRM*, FMT*, RRT*)
```

information) may be from successful motion plans, previous trajectories in the state space, human demonstration, or other sources that provide insight into how the system operates. In this work we use each of these data sources (Section IV), though, when available, optimal solutions to previous motion planning problems are preferred since these will intuitively provide the most insight into the optimal motion planning problem. In order to generate the required breadth of data, (in this work, on the order of one hundred thousand sample points), we leverage GPU-accelerated, approximate motion planning algorithms to generate plans quickly [26]. The data is then processed into the state of the robot and the conditioning variables. In particular, these conditioning variables (Line 2) contain information about the state of the problem, such as workspace information (e.g., obstacles) or the initial state and goal region. The CVAE is then trained in Line 3 according to the following process, with the goal of learning the internal representation of the system conditioned on external properties of the problem (which may inform where in the state space the system will operate, adaptively to a problem).

Let $x$ denote a continuous random variable, $z$ denote the latent variable and $y$ denote the conditioning variable, such that $z$ is drawn from the prior $p_{\theta^*}(z \mid y)$ and $x$ is drawn from the conditional distribution $p_{\theta^*}(x \mid z, y)$. In this case, these distributions are parametric, and $\theta^*$ denotes the true set of parameters. Generally, we write $p_\theta$ to denote parameterized distributions. In our work, $x$ is a sampled state, and $y$ is a set of environmental or problem-specific factors. We aim to construct the distribution $p_{\theta^*}(x \mid y)$. The outline of the training process is outlined in Fig. 2a. We fix the prior distribution $p_\theta(z \mid y)$ to be the isotropic normal distribution $\mathcal{N}(0, I)$, and seek a deterministic function, $f$, that maps this distribution to $p_\theta(x \mid z, y)$ (typically, called a decoder). To learn the decoder, we rely on an encoder of the form $q_\phi(z \mid x, y)$ (where $\phi$ denotes distribution parameters). In practice, this is a deterministic function that maps samples to a mean ($\mu$) and variance ($\Sigma$) in the latent space.

The estimation of the set of the distribution parameters $\theta$ through maximum likelihood methods is generally an intractable problem, and so the variational lower bound (also



(b) Sampling

(a) Training

Fig. 2: Conditional Variational Autoencoder (CVAE) setup [27]. In the context of this work, $x$ represents training states, $y$ the conditioning variable (possibly initial state, goal state, and workspace information), and $z$ the latent state. The decoder (b) is used online to project conditioned latent samples into our distribution.

referred to as the Evidence Lower Bound or ELBO),

$$\log p_\theta(x \mid y) \geq -D_{KL}(q_\phi(z \mid x, y) \,\|\, p_\theta(z \mid y)) + \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x \mid z, y)]$$

is optimized instead, where $D_{KL}(\cdot)$ denotes the KL divergence. This lower bound can then be approximated with Monte Carlo methods, giving the empirical lower bound,

$$\log p_\theta(x) \geq -D_{KL}(q_\phi(z \mid x, y) \,\|\, p_\theta(z \mid y)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x \mid z, y^{(l)}). \quad (1)$$

Using the "reparameterization trick," (covered in detail in [25]), this estimator can be maximized with stochastic optimization techniques standard in machine learning.

The standard construction of the CVAE consists of neural networks for the encoder $q_\phi(z \mid x, y)$ and the decoder $p_\theta(x \mid z, y)$. The encoder and decoder are trained jointly using the reconstruction error (see [27]). Once trained, the decoder allows us to approximately generate samples from $p_{\theta^*}(x \mid y)$ by simply sampling from the normal distribution of the latent variable (see Fig. 2b). While one iteration of this offline phase is often sufficient, with problems that are expensive to solve and thus expensive to acquire data for, the entire methodology may be performed iteratively. Thus, a partially trained CVAE may generate samples that result in better planning performance, allowing more, high-quality data and subsequently allowing the CVAE to be further trained.

The online phase of the methodology begins with a planning problem, Line 4, defined by the tuple $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, which is formed into a conditioning variable $y$ in Line 5. For example, the initial state, the goal state, or workspace obstacles encoded in an occupancy grid. With this in hand, we now generate samples by sampling the latent space as $\mathcal{N}(0, I)$, conditioning on $y$, and projecting these samples to the state space through the decoder network (Line 6). In order to maintain the ability of SBMP algorithms to represent the true state space with arbitrarily high fidelity, and thus maintain the theoretical guarantees of SBMP algorithms (see Remark 1), we also sample from an auxiliary sampler, in our case a uniform sampler. We denote the fraction of learned samples as $\lambda$, i.e., we generate $\lambda N$ samples from the learned sampler and $(1 - \lambda)N$ from the auxiliary sampler. We have found through experimentation that $\lambda = 0.5$ represents the best balance between leveraging the learned sample regions and ensuring full coverage of the state space. Finally, in Line 8, we use these samples to seed a SBMP algorithm, such as PRM$^*$, FMT$^*$, or RRT$^*$, and solve the planning problem. This methodology is applied to a variety of problems with varying state space dimensionality, constraints, and training data-generation approaches in the following section.

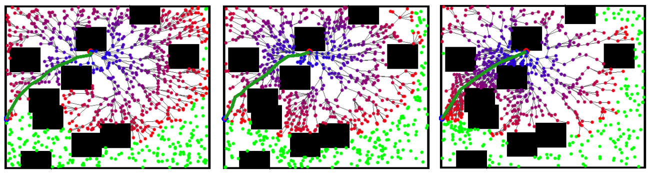
**Remark 1** (Probabilistic Completeness and Asymptotic Optimality)**.** *Note that the theoretical guarantees of probabilistic completeness and asymptotic optimality from [23], [24], and [22] hold for this method by adjusting any references to $n$ (the number of samples) to $(1 - \lambda)N$ (the number of uniform samples in our methodology). This result is detailed in Appendix D of [23] and Section 5.3 of [24], which show that adding samples can only improve the solution or lower the dispersion (which the theoretical results are based on), respectively.*

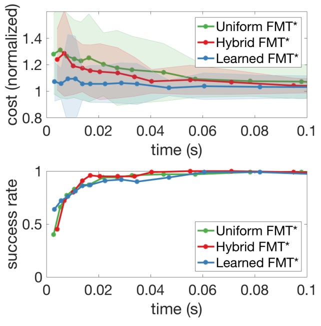
## IV. Numerical Experiments

To demonstrate the performance and generality of learning sample distributions, this section shows several numerical experiments with a variety of robotic systems. The results in Section IV-A were implemented in MATLAB with the Fast Marching Tree (FMT$^*$) and Batch Informed Trees (BIT$^*$) algorithms [23], [3], while the remainder of the results were implemented in CUDA C with a GPU version of the Probabilistic Roadmap (PRM$^*$) algorithm (for convergence plots) and the Group Marching Tree (GMT$^*$) algorithm (to generate training data) [22], [26]. The CVAE was implemented in TensorFlow. The simulations were then run on a Unix system with a 3.4 GHz CPU and an NVIDIA GeForce GTX 1080 Ti GPU. Example code and the network architecture may be found at `https://github.com/StanfordASL/LearnedSamplingDistributions`. We begin with a simple geometric planning problem in which we show our method performs as well as or better than state of the art approaches. We also note that these state of the art approaches are less general than the method we present in this paper, and tuned well towards these geometric problems. We then demonstrate the benefits of learning distributions for a high-dimensional spacecraft system and a dynamical system conditioned on workspace obstacle information. These results are examined conceptually as well as quantitatively in terms of convergence, finding an order of magnitude improvement in success rate and cost. Finally, we apply the method to two very complex systems, a humanoid walker and a multirobot system (trained from human demonstration), and conceptually discuss their resulting distributions. Note sample generation time is included in runtime, but generally accounts for only a fraction of the total runtime–generating thousands of samples takes a only few milliseconds.

### A. Geometric Planning Comparisons

While this methodology is very general and can be applied to complex systems, we first show the methodology performs


(a) Uniform FMT$^*$ (b) Hybrid FMT$^*$ (c) Learned FMT$^*$


(d) Convergence for FMT$^*$

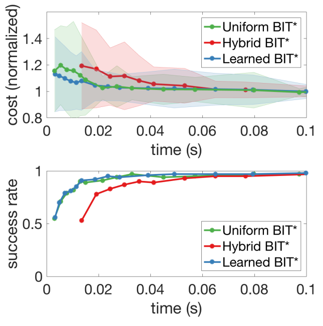

(e) Convergence for BIT$^*$

Fig. 3: (3a-3c) Solutions to the geometric planning problem with different sample distributions and (3d-3e) convergence results for sample distrubtions with FMT$^*$ and BIT$^*$ (results averaged over 100 runs, standard deviation shaded, and $\lambda = 0.5$).

well for a simple, geometric problem. All problems are created with randomly generated initial states, goal states, and 10 cube obstacles, as shown in Fig. 3. The learned sample distributions were conditioned on all the problem information (initial and goal states and obstacles), and trained over successful motion plans.

For this problem, we make comparisons to a non-uniform sampling strategy and combine our methodology with an exploration-guided non-uniform sampling algorithm. Specifically, we consider the hybrid sampling strategy proposed in [28], and BIT$^*$ [3]. The hybrid sampling approach uses uniform samples, Gaussian samples, and bridge samples to create a distribution favoring narrow passageways and regions nearby obstacles. BIT$^*$ uses successive batches of samples to iteratively refine a tree, leveraging solutions from previous batches to selectively sample only states that can improve the solution.

The results of these comparisons are shown in Fig. 3d. The first comparison shows each strategy with FMT$^*$ [23]. We find that each strategy performs nearly equivalently in terms of finding a solution, but the learned strategy finds significantly better solutions in the same amount of time. In fact, the learned sampling strategy finds within 5% of the best solution almost immediately, instead of converging to it. The results show less of a performance gap with BIT$^*$, though the learned strategy continues to perform at least as well as the others. The delayed start of the hybrid convergence is only due to the time required to generate samples, which for BIT$^*$ was implemented here by rejection.

## B. Spacecraft Debris Recovery

The second numerical experiment considered is a simplified spacecraft debris recovery problem, whereby a cube shaped spacecraft with 3D double integrator dynamics ($\ddot{x} = u$, no rotations) and a pair of 3 DoF kinematic arms (assumed to be much less massive than the spacecraft body), for a total of 12 dimensions, must maneuver from an initial state, through a cluttered asteroid field, to recover debris between its end effectors. The cost is set as a mixed time/quadratic control effort penalty with an additional term for joint angle movement in the kinematic arms. The initial and goal states were randomly generated, as were the asteroids (i.e., obstacles). Fig. 4 shows an example problem and the spacecraft setup. The CVAE was conditioned on the initial state and debris location, and trained with successful motion plans.

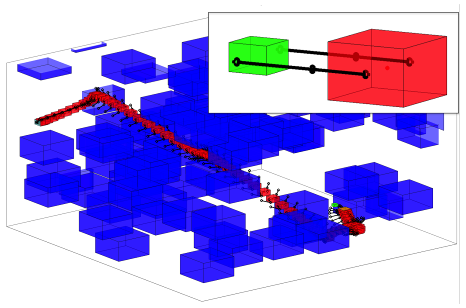


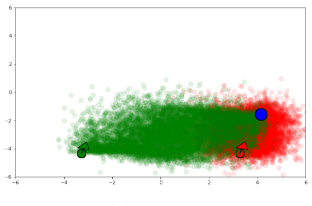
Fig. 4: An example spacecraft debris recovery problem, whereby the spacecraft must maneuver from an initial state to recover debris between its end effectors. The spacecraft is modeled as a double integrator with a pair of 3 DoF kinematic arms.

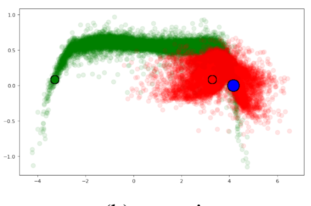
The resulting learned distributions are shown in Fig. 5. Fig. 5a shows the learned distribution of the $x$ and $y$ positions for two problems. The distribution resembles an ellipsoid connecting the initial and goal states, with some spread in the minor axis to account for potential obstacles in the trajectory and a slight skew in the direction of the initial velocity–we note the similarity to the sample distributions found after exploration in BIT$^*$ [3]. Fig. 5b shows the learned distribution of $x$ and $\dot{x}$, i.e., a phase portrait of the $x$ dimension. The green distribution favors velocities such that any sample flows from the initial to the goal state, first accelerating near the maximum sampled velocity ($\dot{x} = 1$), and then maintaining the velocity until nearby the goal. The red distribution, whose initial position is much closer to the goal region has a much larger spread, favoring samples that move towards the goal in all directions. Finally, the learned distributions for a single arm are shown in Figs. 5c-5d. The angle distributions demonstrate the arm movement should be kept to a minimum, by holding one dimension fixed to a few values only. In the problem setup, the arms have significantly less impact on obstacle avoidance, but can incur a large cost for movement, which is reflected in the distributions.

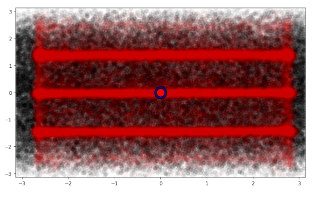
Fig. 6 shows the convergence of the methods in time. The learned sampling distribution outperforms the uniform by approximately an order of magnitude in finding solutions when they exist. The cost convergence curve is even more extreme; the learned sample converges to a few percentage from optimal almost immediately, while even after 10,000 samples, the uniform distribution is still more than 60% from optimal. This immediate convergence is similar to what was observed in the geometric planning problem and the narrow passage problem (in the following section). We also note the variance is smaller for the learned distributions.

## C. Workspace Learning

Our next problem, shown in Figs. 1 and 7, was loosely inspired by the narrow passage problems in [9], demonstrates


(a) $x$ vs $y$


(b) $x$ vs $\dot{x}$


(c) $\alpha$ vs $\beta_1$

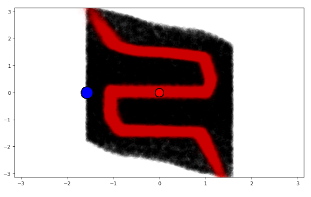

(d) $\beta_1$ vs $\beta_2$

Fig. 5: Spacecraft learned distributions for various dimensions. The learned distributions are shown in red and green, the goal state is shown in blue, and the initial training distributions are shown in black (when displayed). (5a) shows the distribution favoring an ellipse connecting the initial and goal states. (5b) shows a phase portrait of the $x$ dimension, where the samples favor velocities towards the goal region. (5c-5d) show distributions for the kinematic arm angles, where it has converged to a few fixed values to sample, thus reducing movement cost ($\alpha$ and $\beta_1$ denote rotation angles at the arm-spacecraft hub and $\beta_2$ denotes the joint angle in the arm).

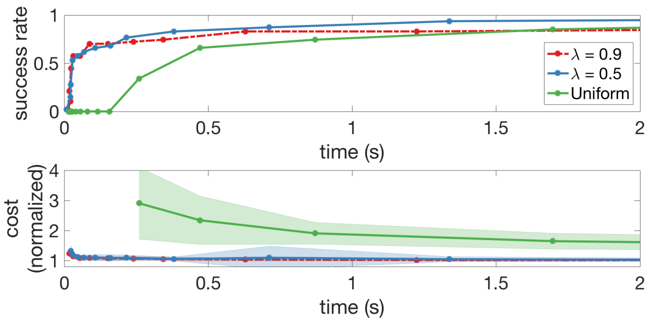


Fig. 6: Convergence results for the spacecraft planning problem. The learned distributions (50% and 90% learned) significantly outperform a uniform distribution (results averaged over 100 runs and the standard deviation shaded).

the ability of the methodology to learn distributions conditioned on workspace information. The problem features a 3D double integrator (6 dimensional state space) operating in an environment with 3 narrow passages. The initial state, goal state, and gap locations are all randomly generated and used to condition the CVAE for each problem (obstacles were passed through an occupancy grid). Fig. 7 shows several problems and their learned distribution; clearly, the CVAE has been able to capture both initial state and goal state biasing, some sense of dynamics, and the obstacle set. The velocity distributions too show the samples effectively favoring movement from the initial to goal state. The convergence results, shown in Fig. 8, demonstrate learned distribution solutions can be found with approximately an order of magnitude fewer samples and converge in cost almost immediately, while the uniform sampling results show the classic convergence curve we may expect.

This method's ability to learn both dynamics and obstacles demonstrates that learning is capable of almost entirely solving some problems. While this would be quite efficient,

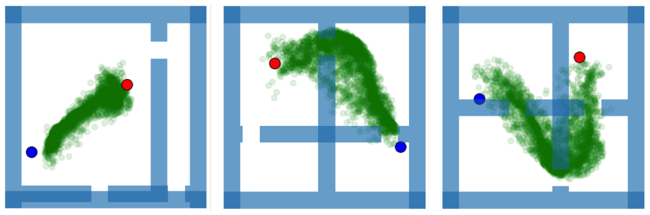


Fig. 7: Example learned distributions for the narrow passage problem, conditioned on the initial state (red), goal state (blue), and the obstacles (through an occupancy grid).

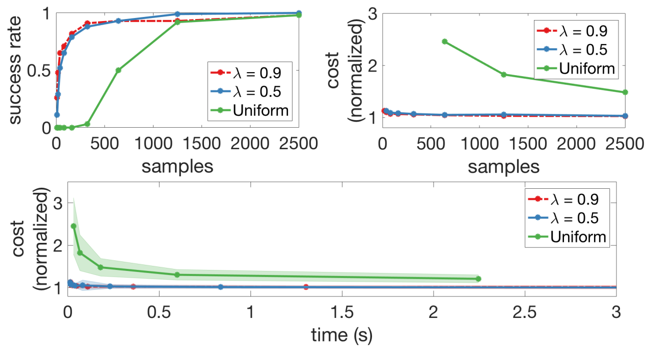


Fig. 8: Convergence results for the narrow passage problem, demonstrating the learned sample distributions (50% and 90% learned) achieve approximately an order of magnitude better performance in terms of success rate, and are able to converge with few samples (results averaged over 100 runs and the standard deviation shaded).

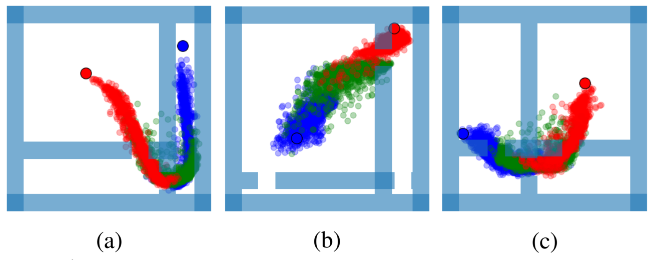
we also found the learned distributions susceptible to failure modes (e.g., cutting corners), which result in infeasible solution trajectories. In our methodology, this is easily handled through the uniform sampling and the guarantees of SBMP. This methodology may thus be thought of as attempting to solve the problem through learning, and accounting for possible errors with a theoretically sound algorithm.

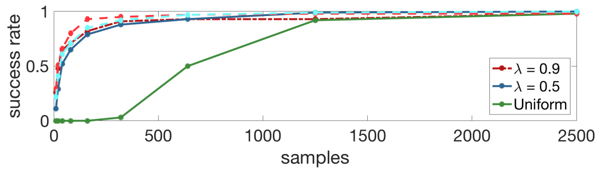
## D. Learning Dependent Sample Sets

To showcase the generality of the learning distributions methodology and its ability to capture arbitrary and complex distributions, we used the same methodology with sets of samples (in this case, solutions that contained three or more samples). We thus are learning not only the distribution for one sample, but three samples with dependency between them (in particular, they should be well dispersed). Fig. 9 shows resulting distributions and the success rate of this method compared to learning only a single sample. As expected, the distributions learn to be well-distributed along the solution trajectory, resulting in higher success rates (e.g., the 90% of learned to uniform increased from 82% to 93% at 100 samples). This result corroborates the findings of [24], which found the primary benefit of low-dispersion sampling is in finding solutions with fewer samples.

## E. Humanoid Walker

We demonstrated our methodology on samples from the `HumanoidFlagrunHarder-v1` task of the Roboschool learning environment [29]. In this task, a humanoid figure is given a goal location in the workspace every 1000 frames, and receives a reward relative to the its distance from this goal. The humanoid policy trained was taken from an example script, in which Proximal Policy Optimization





(d) Convergence

Fig. 9: (9a-9c) Learned distributions of multiple, dependent samples drawn together (each one in red, blue and green), effectively enforcing some dispersion between them. (9d) Success rates for single sample distributions and multi-sample distributions (lighter colors denote the multi-sample counterparts).

[29] was used to train a feedforward neural network policy parameterization. The state space of this system is 44-dimensional, corresponding to joint angles, as well as the position of the desired goal, and binary variables indicating whether each foot was in contact with the ground. Fig. 10 shows four samples from our CVAE (left) a sample with randomly sampled joint angles as comparison (right). One may note that in all of our samples, the humanoid's arms are raised forward and in front of its body. In the learning environment, small blocks are intermittently thrown at the humanoid so that the learned policy is more robust to disturbances. In response, the learned policy keeps its arms extended to provide stability. The samples are in the full state space, so joint velocities are also sampled, but those are not visualized here.

Note, we only examine our methodology qualitatively in this section due to the difficulty in making convergence comparisons to uniform sampling; uniform sampling would require orders of magnitude more samples and computation time than considered in previous problems and would not be an approach considered in practice. Specialized techniques do exist towards selecting samples for humanoid robotic systems, however these have required expert tuning to specific problems, are more computationally intensive, and generally more conservative [30], [19], [31]. Many of these approaches are based on sampling configurations that are statically balanced, with little consideration for sampling joint velocities. Our approach generates samples from a humanoid robot successfully running, implying that these states are dynamically stable since the system was trained with added disturbances. The generated samples are dynamic, and thus can yield substantially lower costs as they may be fluidly connected.

### F. Multirobot and Human Demonstration

Finally, we demonstrate the methodology in a multi-robot problem trained through human demonstration. The problem is set up as two cars completing a lane changing maneuver (Fig. 11a), with a total of 22 states, as well as a constraint on the two cars colliding and a preference towards the

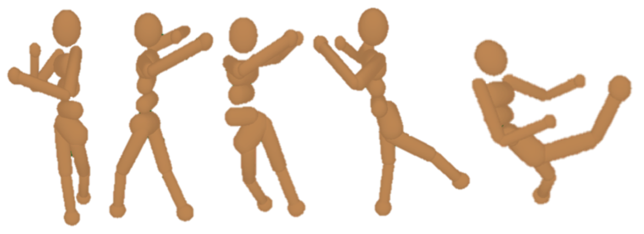


Fig. 10: Four samples from the learned distribution for a 44 dimensional humanoid model (left) and one drawn from a uniform random distribution (right). Note the learned samples are in robust (arms up), upright poses, thus allowing momentum to be maintained.

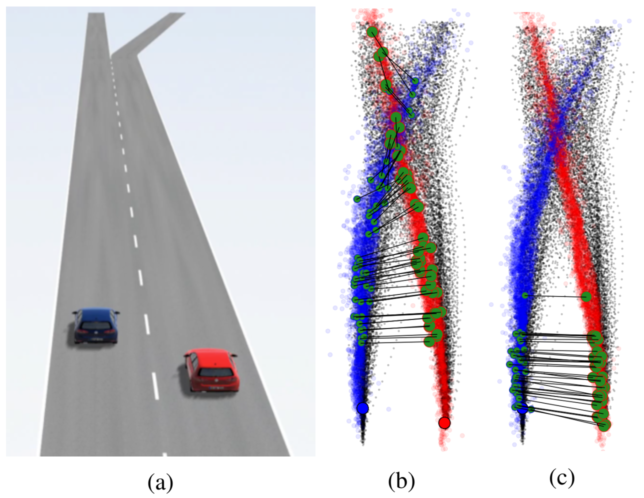


Fig. 11: (11a) Setup for lane change problem, generated from human demonstration, where the red and blue car must switch lanes and make their respective exits. (11b-11c) The training dataset is visualized in black, overlayed with learned distributions for the blue and red initial car states. Displayed are the $x$ and $y$ positions of each car at several samples. Initially, the red car is behind the blue car, but at a higher velocity and thus eventually passes the blue car in most samples (the red car is leading samples visualized in (11b) and tailing in (11c)). Note each sample is not in self collision.

cars remaining in their lanes when not changing lanes. The data was collected from human demonstration on a two person driving simulator. The resulting learned distribution is visualized for a single initial state in Fig. 11b and overlayed on the total dataset. Again, we only examine this problem qualitatively due to the computational challenges of solving the same problem with uniform samples. The learned distribution effectively encapsulates the necessary factors for a successful motion plan: the collision constraint, the preference towards the center of lanes, the preference to maintain forward velocity, and the choice of states applicable to a lane change maneuver.

## V. Conclusions

*Conclusions:* In this paper we have presented a methodology to bias samples in the state space for sampling-based motion planning algorithms. In particular, we have used a conditional variational autoencoder to learn subspaces of valid or desirable states, conditioned on problem details such as obstacles. We have compared our methodology to several state of the art methods for sample biasing, and have demonstrated it on multiple systems, showing approximately an order of magnitude improvement in cost and success rate over uniform sampling. This learning-based approach is promising due to its generality and extensibility, as it can be applied to any system and can leverage any problem information available. Its ability to automatically discover useful representations for motion planning (as seen by the near immediate convergence) is similar to recent results from deep learning in the computer vision community, which require less human intuition and handcrafting, and exhibit superior performance.

*Future Work:* There are many possible avenues open for future research. One promising extension is the incorporation of semantic workspace information through the conditioning variable. These semantic mappings show promise towards allowing mobile robots to better understand task specifications and interact with humans [32]. Another promising extension builds upon recent work demonstrating the favorable theoretical properties and improved performance of non-independent samples [24]. An approach to reducing the independence of the samples was presented in Section IV-C, but extensions beyond this exist, including generating large sample sets (e.g., $> 1000$). Lastly, for systems with constraints that force valid configurations to lie on zero-measure manifolds, recent work has focused on projective methods [33]. Our methodology, while not capable of sampling directly on this manifold, can easily learn to sample near it, and therefore, potentially dramatically improve the performance of these methods.

*Application in Practice:* Note again, the goal of this work is to compute a distribution representing promising regions (i.e., regions where optimal motion plans are likely to be found) through a learned latent representation of the system conditioned on the planning problem. During the offline CVAE training phase, we recommend training from optimal motion plans to best demonstrate promising regions. We generally train on the order of one hundred thousand motion plans, the generation of which can be accelerated with approximately optimal, GPU-based planning algorithms [26]. To form the conditional variable, we recommend including all available problem information, particularly the initial state and goal state. If available, workspace obstacles can be easily included through occupancy grids; we note however that obstacles can be included in any form, as the neural network representation for the CVAE has the potential to arbitrarily project these obstacles as necessary. Multiple *dependent* samples can also be generated at once (Section IV-D); we found even as few as three resulted in marked increases in success rate. Finally, in the online phase of the algorithm, we draw a combination of learned and uniform samples to ensure good state space coverage; we found a 50-50 split to be most effective.

## Acknowledgment

This work was supported by a Qualcomm Innovation Fellowship and by NASA under the Space Technology Research Grants Program, Grant NNX12AQ43G. James Harrison was supported in part by the Stanford Graduate Fellowship and the National Sciences and Engineering Research Council (NSERC). The authors also wish to thank Edward Schmerling for helpful discussions.

## References

[1] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *NIPS*, 2015.

[2] D. Hsu, J.-C. Latombe, and H. Kurniawati, “On the probabilistic foundations of probabilistic roadmap planning,” *IJRR*, 2006.

[3] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in *IEEE ICRA*, 2015.

[4] H. Kurniawati and D. Hsu, “Workspace importance sampling for probabilistic roadmap planning,” in *IEEE IROS*, 2004.

[5] J. P. Van den Berg and M. H. Overmars, “Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners,” *IJRR*, 2005.

[6] Y. Yang and O. Brock, “Adapting the sampling distribution in PRM planners based on an approximated medial axis,” in *IEEE ICRA*, 2004.

[7] R. Geraerts and M. H. Overmars, “Sampling techniques for probabilistic roadmap planners,” *IAS*, 2004.

[8] B. Burns and O. Brock, “Toward optimal configuration space sampling.” in *RSS*, 2005.

[9] M. Zucker, J. Kuffner, and J. A. Bagnell, “Adaptive workspace biasing for sampling-based planners,” in *IEEE ICRA*, 2008.

[10] D. Berenson, P. Abbeel, and K. Goldberg, “A robot path planning framework that learns from experience,” in *IEEE ICRA*, 2012.

[11] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, “Experience-based planning with sparse roadmap spanners,” in *IEEE ICRA*, 2015.

[12] Y. Li and K. E. Bekris, “Learning approximate cost-to-go metrics to improve sampling-based motion planning,” in *IEEE ICRA*, 2011.

[13] I. Baldwin and P. Newman, “Non-parametric learning for natural plan generation,” in *IEEE IROS*, 2010.

[14] G. Ye and R. Alterovitz, “Demonstration-guided motion planning,” in *ISRR*, 2015.

[15] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE TPAMI*, 2013.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *JMLR*, 2016.

[17] I. Havoutis and S. Ramamoorthy, “Motion synthesis through randomized exploration on submanifolds of configuration space,” in *Robot Soccer World Cup*, 2009.

[18] P. Dollár, V. Rabaud, and S. Belongie, “Non-isometric manifold learning: Analysis and an algorithm,” in *ICML*, 2007.

[19] N. Chen, M. Karl, and P. van der Smagt, “Dynamic movement primitives in latent space of time-dependent variational autoencoders,” in *IEEE-RAS HUMANOIDS*, 2016.

[20] E. Schmerling, L. Janson, and M. Pavone, “Optimal sampling-based motion planning under differential constraints: the driftless case,” *IEEE ICRA*, 2015.

[21] S. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[22] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *IJRR*, 2011.

[23] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast Marching Tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *IJRR*, 2015.

[24] L. Janson, B. Ichter, and M. Pavone, “Deterministic sampling-based motion planning: Optimality, complexity, and performance,” *IJRR*, 2017.

[25] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[26] B. Ichter, E. Schmerling, and M. Pavone, “Group Marching Tree: Sampling-based approximately optimal motion planning on GPUs,” in *IEEE IRC*, 2017.

[27] C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908*, 2016.

[28] D. Hsu, G. Sánchez-Ante, and Z. Sun, “Hybrid PRM sampling with a cost-sensitive adaptive strategy,” in *IEEE ICRA*, 2005.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[30] S. Dalibard, A. El Khoury, F. Lamiraux, A. Nakhaei, M. Taïx, and J.-P. Laumond, “Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach,” *IJRR*, 2013.

[31] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, “Using motion primitives in probabilistic sample-based planning for humanoid robots,” in *WAFR*, 2008.

[32] I. Kostavelis and A. Gasteratos, “Semantic mapping for mobile robotics tasks: A survey,” *Robotics and Autonomous Systems*, 2015.

[33] L. Jaillet and J. M. Porta, “Path planning under kinematic constraints by rapidly exploring manifolds,” *IEEE TRO*, 2013.