Fast Swept Volume Estimation with Deep Learning

Hao-Tien Lewis Chiang[1,2], Aleksandra Faust[2], Satomi Sugaya [1], Lydia Tapia[1]

[1] Department of Computer Science, University of New Mexico, MSC01 11301
University of New Mexico, Albuquerque, NM 87131, USA. E-mail:
lewispro@unm.edu, satomi@unm.edu and tapia@cs.unm.edu.
[2] Google Brain, Mountain View, CA 94043, USA. E-mail: lewispro@google.com and
faust@google.com.

Abstract. Swept volume, the volume displaced by a moving object, is an ideal distance metric for sampling-based motion planning because it directly correlates to the amount of motion between two states. However, even approximate algorithms are computationally prohibitive. Our fundamental approach is the application of deep learning to efficiently estimate swept volume computation within a 5%-10% error for all robots tested, from rigid bodies to manipulators. However, even inference via the trained network can be computationally costly given the often hundreds of thousands of computations required by sampling-based motion planning. To address this, we demonstrate an efficient hierarchal approach for applying our trained estimator. This approach first pre-filters samples using a weighted Euclidean estimator trained via swept volume. Then, it selectively applies the deep neural network estimator. The first estimator, although less accurate, has metric space properties. The second estimator is a high-fidelity unbiased estimator without metric space properties. We integrate the hierarchical selection approach in both roadmap-based and a tree-based sampling motion planners. Empirical evaluation on the robot set demonstrates that hierarchal application of the metrics yields up to 5000 times faster planning than state of the art swept volume approximation and up to five times higher probability of finding a collision-free trajectory under a fixed time budget than the traditional Euclidean metric.

1 Introduction

Illustrated in Figure 1, swept volume, $\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2)$, is the measure of the volume displaced by an object moving between two configurations, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, [1, 2]. Swept volume computation has proven useful in several applications including machining verification, geometric modeling, collision detection, mechanical assembly, ergonomic studies, and product lifecycle management [3]. Additionally, swept volume was identified as an ideal sampling-based planning distance metric because it directly correlates to amount of motion required to transition between two collision-free configurations [4]. Yet, swept volume has not been commonly

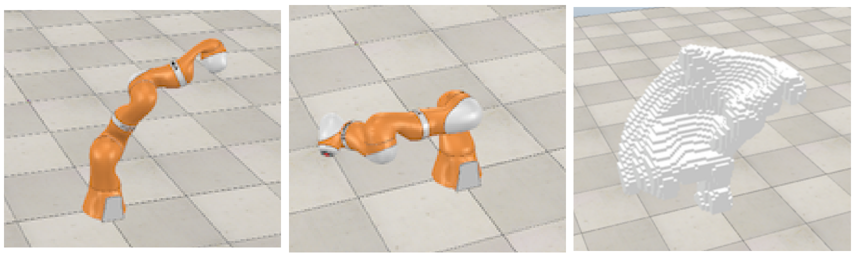

(a) Start configuration (b) End configuration (c) Swept volume

Fig. 1: Example *start* (a) and *end* (b) configurations and the corresponding volume swept (c) for a Kuka LBR IIWA 14 R820 manipulator.

used in sampling-based planning due to the complexities in its computation. The problem lies in the intractability of exact swept volume computation, due to the frequent requirement of the construction of a complex non-linear geometry. As a result, most algorithms focus on generating approximations [5], such as occupation grid-based and boundary-based methods [5, 3, 6, 7, 1]. However, even these approximations are too computationally expensive to be used as a practical distance metric in sampling-based motion planners [4, 8] since distance metric calls are one of the most common operations [9].

Sampling-based planners select a subset of configurations to attempt expensive local planning operations that make connections in roadmap-based, e.g., a probabilistic roadmap method (PRM) [10], or tree extensions in tree-based planners, e.g., a rapidly-exploring random tree (RRT) [11]. The configurations are typically selected w.r.t. some distance metric. Intuitively, a good metric limits the local planning operations to those most likely to succeed, i.e., lowest collision probability between the two configurations without prior knowledge of obstacle distribution. Yet, the commonly chosen metrics suffer from several issues. For example, the configuration space Euclidean distance metric does not represent collision probability well [4, 9], and the weighted Euclidean distance metric is difficult to tune [9].

Our fundamental approach lies in the ability of a deep neural network (DNN) to approximate swept volume. Trained off-line with swept volume examples for a given robot in an obstacle-free space, the DNN captures the complex and nonlinear relationship between trajectories in the robot's C-space. In this paper, we show that successful learning is feasible since DNNs have been proven to be able to approximate any continuous bounded function [12], and swept volume possesses these properties in a finite C-space. Also, through empirical evaluation on a variety of robot types, from a rigid body to manipulators, we find that the DNN estimators are able to capture swept volumes within a 5-10% error at speeds 3500 to 5000 times faster than traditionally used approximate swept volume computation.

Despite the increased speedup by using a DNN for swept volume estimation, inference via the DNN can remain prohibitive due to the often hundreds of thousands or more of inference calls required in difficult sampling-based motion planning problems. In addition, $\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ does not form a metric space as it does not satisfy the triangle inequality. This prevents utilization of efficient nearest neighbor data structures such as GNAT [13] common for sampling-based planners. Therefore, we also propose an efficient hierarchal approach for applying our trained estimator, Hierarchical Neighbor Search (HNS). Specifically, during nearest neighbor selection in sampling-based planning when comparison via metric is applied, we pre-filter configurations using a weighted Euclidean estimator trained via swept volume. This first estimator is a single layer neural network that has metrics properties but less accuracy. Then, we can selectively apply the DNN, a high-fidelity estimator without metric properties. The hierarchical combination in HNS demonstrates both metric properties and high-fidelity, thus enabling efficient selection of configurations with low swept volume using existing nearest neighbor data structures that require metric space properties.

We achieve feasible swept volume integration with sampling-based planning in three steps. First, we generate the training data using an occupation grid-based [6] swept volume approximation. This is by far the most computationally intensive step, as the swept volume is computed for random configuration pairs. Second, we train the models needed for swept volume estimation. The training is also computationally intensive, but less so than the data generation. Both steps are done once, prior to planning. Lastly, sampling-based planners such as RRT and PRM use nearest neighbors computed with the metrics to attempt the connections between configurations. We use an efficient nearest neighbor data structure, GNAT [13], with the metric estimators.

Evaluation of the metrics and HNS on three robots, a 6 Degree Of Freedom (DOF) rigid body, a 15 DOF fixed-based planar manipulator, and a 7 DOF Kuka LBR IIWA 14 R820 in a cluttered, a narrow corridor, and a real-application inspired environment. Two popular sampling-based planners, PRM and RRT are integrated with our metrics. Planners that use HNS are: 1) up to five times more likely to identify a collision-free path on a fixed time budget and 2) able to return paths with a smaller swept volume. These advantages are consistent for all three robots and are particularly significant for robots with a highly articulated body. The video in the supplementary material contains visualization of the simulations.

The contributions of this paper include learning a high-precision swept volume estimator, datasets for learning, and demonstrations of the metrics within planning. Specifically, we highlight the utility of high-fidelity DNN swept volume models and efficient hierarchical combination of learned metrics, HNS, for sample neighbor selection during roadmap connection and tree expansion. The weighted Euclidean metric and datasets are publicly available for the three robots used in this paper, and can be readily used by other researchers without any modifications. In addition, this paper might be of interest to the larger motion planning community as an example of a machine learning's role within the motion planning. It is a proof of concept that a computation of a highly complex and non-linear, yet Lipschitz continuous and bounded, metric can be learned ahead of time off-line from examples, reducing the complexity of planning.

2 Related Work

Modern approximate swept volume algorithms can be roughly classified as occupation grid-based and boundary-based methods. Occupation grid-based approaches decompose the workspace, e.g., into voxels, in order to record the robot's occupation in the workspace as it executes a trajectory [3, 6]. The resulting approximation has a resolution-tunable accuracy and is conservative, which can be critical for applications such as collision avoidance [14]. The boundary-based methods extract the boundary surface [7, 5, 1]. Despite more than four decades of study, swept volume computation is still too slow to be used online by sampling-based motion planners [4, 8]

Swept volume has been used in motion planning in various ways. In [14], swept volume of a bipedal robot for 276 step patterns are computed offline and queried online by an RRT-based planner to speed up collision detection for robot footstep planning. Similarly, in this paper we compute swept volume approximations offline. However, our learned estimators can generalize to unseen configuration pairs. Swept volume had also been used directly as a distance metric as a comparison method in [8]. However, due to the exceedingly high swept volume computation cost, the performance is reported to be orders of magnitude worse than weighted euclidean distance metrics.

A distance metric that accurately predicts a local planner's success rate is critical for sampling-based motion planners [15]. On the other hand, distance metric calculations also need to be fast since they are one of the most numerous sampling-based planners operations [9]. Carefully tuned weighted Euclidean distance metrics have been empirically shown to outperform other metrics [9]. This conclusion is echoed in [16] where a weighted Euclidean distance metric is tuned to approximate swept volume. However, a weighted Euclidean distance metric may not be expressive enough to approximate swept volume as it is clearly nonlinear, i.e., each joint DOF affects one another in an articulated body.

Machine learning has been used to learn distance metrics for kinodynamic robots [17, 18]. In such systems, the design goal of distance metrics is often quite different from planning in C-space due to the constrained reachability. Good distance metrics typically approximate the minimum time to reach between states [17]. In [17], regression learning algorithms are trained offline to approximate the time to reach between states. The training data is generated by a near-optimal controller for unicycle robots. A RRT-based planner then uses the learned distance metric during online planning. A similar method replaces the near-optimal controller with an indirect controller to learn both the time to reach and control inputs [18]. These methods differ from ours, in that our methods identify neighboring configurations that are likely to succeed in the connect or extend operations due to expected distance of a trajectory, while distance metrics in [18, 17] approximate minimum time to reach. Another example of integration between deep machine learning and sampling-based planning is PRM-RL [19]. Similar to our method, it uses an offline, once-per robot training to connect PRM nodes that are most likely to succeed. Unlike our learned distance estimators, PRM-RL learns the local planner for a physical robot moving via sensor information. Therefore, nodes that are most likely to succeed are connected.

3 Methods

Swept volume, $\mathcal{SV} : \mathbb{R}^{2d_f} \to \mathbb{R}$, for a trajectory in C-space is

$$\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2) = \| \cup_{t \in [0,1]} \mathcal{V}((1-t)\boldsymbol{c_1} + t\boldsymbol{c_2}) \|, \tag{1}$$

where $\boldsymbol{c_1}, \boldsymbol{c_2} \in \mathbb{R}^{d_f}$ are the start and end configurations of a robot with d_f degrees of freedom, and $\mathcal{V}(\boldsymbol{c})$ is the workspace occupied by the robot in configuration $\boldsymbol{c}$. We consider the trajectory between $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ to be a straight line in C-space. $\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ can be highly complex and nonlinear due to rotational degrees of freedoms, especially in cases where the robot has an articulated body.

Swept volume estimator models are the trained offline in three steps: 1) $\mathcal{SV}$ training dataset generation described in Section 3.1, 2) learning a weighted Euclidean distance metric, $\mathcal{D}^{\text{sv}}_{\text{we}}$, using a single layer network in Section 3.2, and 3) training a deep swept volume estimator in Section 3.3. After the off-line learning, Section 3.4 describes how HNS efficiently combines the two estimators into a hierarchical neighbor selector that selects the most promising nodes in sampling-based planning.

3.1 Training Dataset Generation

The training data $(\boldsymbol{X}, \boldsymbol{y})$ of size n, where $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^\top$, and each training sample $\boldsymbol{x}_i = [\boldsymbol{c}_{i,1}\, \boldsymbol{c}_{i,2}]$ consists of two uniformly-randomly sampled points from the configuration space. The ground truth labels, $\boldsymbol{y}$, match swept volume between two corresponding configurations with respect to the straight line planner. Since $\mathcal{SV}$ is only related to the kinematics of the robot and independent to the environments it operates in, we do not consider obstacles during the generation of training data. Ideally, the labels should be computed with (1), but computing the exact $\mathcal{SV}$ is intractable. Instead, we use labels,

$$\boldsymbol{y} = [y_1, \cdots, y_n]^\top = [\widetilde{\mathcal{SV}}(\boldsymbol{c}_{1,1}, \boldsymbol{c}_{1,2}), \cdots, \widetilde{\mathcal{SV}}(\boldsymbol{c}_{n,1}, \boldsymbol{c}_{n,2})]^\top, \tag{2}$$

approximated with a state of the art octree-based swept volume algorithm [6], where the robot trajectory is represented by N_{lerp} intermediate C-space configurations. The details of the octree-based algorithm are in the supplementary material and the training datasets are available from our website [20].

3.2 Training Weighted Euclidean Distance Estimator, $\mathcal{D}^{\text{sv}}_{\text{we}}$

Weighted Euclidean distance metric, $\mathrm{d}_{\boldsymbol{w}}(\boldsymbol{c_1}, \boldsymbol{c_2}) = \sqrt{\sum_{j=1}^{d_f} w_j (c_{1,j} - c_{2,j})^2}$, often requires manual tuning of the vector $\boldsymbol{w} \in \mathbb{R}^{d_f}$.

We learn the weights, $\boldsymbol{w}^*_{\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}}$, with a single layer network, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$, that models the weighted Euclidean distance metric, $\mathrm{d}_{\boldsymbol{w}}(\boldsymbol{c_1}, \boldsymbol{c_2})$, w.r.t. the training dataset (2),

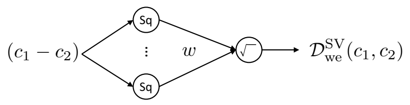
$$\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}(\boldsymbol{w}^*_{\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}}, \boldsymbol{c_1}, \boldsymbol{c_2}) = \mathrm{d}_{\boldsymbol{w*}}(\boldsymbol{c_1}, \boldsymbol{c_2}).$$

The details of the network are depicted in Figure 2(a). We use a stochastic gradient descent optimizer to find $\boldsymbol{w}^*_{\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}}$ that minimizes L_2 loss

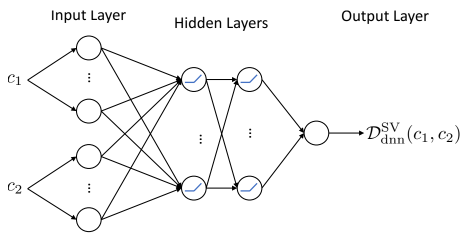
$$\boldsymbol{w}^*_{\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}} = \arg min_{\boldsymbol{w}} \sum_{i=1}^{n} (\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}(\boldsymbol{w}, \boldsymbol{c}_{1,i}, \boldsymbol{c}_{2,i}) - \widetilde{\mathcal{SV}}(\boldsymbol{c}_{1,i}, \boldsymbol{c}_{2,i}))^2. \tag{3}$$

The network is trained once per robot, and like the analytical representation the network, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$, forms a metric space when the weights are positive.

3.3 Training Deep Swept Volume Distance Estimator, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$



(a) Weighted Euclidean Metric model, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$



(b) Deep Swept Volume Model, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$

Fig. 2: Neural network architectures used to estimate $\widetilde{\mathcal{SV}}(\boldsymbol{c}_1, \boldsymbol{c}_2)$. $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the start and end configurations of the trajectory, respectively. (a) Weighted Euclidean Metric model, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}(\boldsymbol{w}, \boldsymbol{c}_1, \boldsymbol{c}_2)$. The input, $\boldsymbol{c_1} - \boldsymbol{c_2}$, is fed to d_f neurons (one per DOF) with a square activation function, i.e., $f(x) = x^2$. The output is the absolute value of the weighted sum of activations. (b) Deep Swept Volume Model, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}(\boldsymbol{w}, \boldsymbol{c}_1, \boldsymbol{c}_2)$. The inputs are $2d_f$. The activation function of the first layer are identities. The input layer is connected to the k hidden layers each with N_i ReLU neurons. The output layer has one neuron corresponding to the swept volume estimate.

The weighted Euclidean distance metric cannot approximate $\widetilde{\mathcal{SV}}$ well because the model may not be expressive enough to capture non-linearities. Therefore, we also use a deep neural network, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$, to learn a non-linear swept volume model. $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ is a fully-connected feed-forward DNN. The inputs, outputs, and the architecture of the $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ are described in Figure 2 (b). The inputs are $2d_f$ input neurons. The first d_f correspond to $\boldsymbol{c}_1$, while the second d_f correspond to $\boldsymbol{c}_2$. The k hidden layers consist of ReLu [21] neurons. Finally, the output is a neuron estimating the swept volume between two configuration points and outputs zero if the network prediction is negative. Stochastic gradient descent backprop finds the weights and biases w.r.t. L_2 loss and the dataset (2),

$$(\boldsymbol{W}^*, \boldsymbol{b}^*)_{\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}} = \arg min_{(\boldsymbol{W}, \boldsymbol{b})} \sum_{i=1}^{n} (\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}((\boldsymbol{W}, \boldsymbol{b}), \boldsymbol{c}_{1,i}, \boldsymbol{c}_{2,i}) - \widetilde{\mathcal{SV}}(\boldsymbol{c}_{1,i}, \boldsymbol{c}_{2,i}))^2. \tag{4}$$

3.4 Hierarchical Neighbor Search, HNS

In this section we propose Hierarchical Neighbor Search, HNS, that combines the trained swept volume estimators introduced above to be used for neighbor selection within sampling-based planning. This hierarchical combination efficiently selects neighbors with low swept volume distance, by first filtering all candidates using the extremely fast learned $\mathcal{D}^{\text{sv}}_{\text{we}}$ and then filtering this smaller subset with $\mathcal{D}^{\text{sv}}_{\text{dnn}}$. In this paper, we implement this filtering by using the k-closest neighbor selection method at each level, but other neighbor connection strategies can be used, such as a distance cutoff [22]. Our implementation first identifies k_c candidate nearest neighbors of configuration $\boldsymbol{c}$ using $\mathcal{D}^{\text{sv}}_{\text{we}}$ (output of the weighted Euclidean metric model). Next, HNS uses the $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ (output of the deep swept volume model) to choose the final $k_{nn} < k_c$ nearest neighbors among the candidates.

Using the following notation to denote selecting nearest neighbors and the corresponding swept volume estimates from a given start configuration $\boldsymbol{c}$ to any element in a given set of configurations X:

$$NN_{\mathcal{D}^{\text{sv}}_{\text{we}}}(X) = \{(\boldsymbol{x}, \mathcal{D}^{\text{sv}}_{\text{we}}(\boldsymbol{w}^*_{\mathcal{D}^{\text{sv}}_{\text{we}}}, \boldsymbol{c}, \boldsymbol{x})) | \boldsymbol{x} \in X\}, \tag{5}$$

and

$$NN_{\mathcal{D}^{\text{sv}}_{\text{dnn}}}(X) = \{(\boldsymbol{x}, \mathcal{D}^{\text{sv}}_{\text{dnn}}((\boldsymbol{W}^*, \boldsymbol{b}^*)_{\mathcal{D}^{\text{sv}}_{\text{dnn}}}, \boldsymbol{c}, \boldsymbol{x})) | \boldsymbol{x} \in X\}, \tag{6}$$

Algorithm 1 depicts the HNS.

Algorithm 1 Hierarchical Neighbor Search for Sampling-based Planning, HNS

Input: $\boldsymbol{w}^*_{\mathcal{D}^{\text{sv}}_{\text{we}}}$: Learned weights in (3).
Input: $(\boldsymbol{W}^*, \boldsymbol{b}^*)_{\mathcal{D}^{\text{sv}}_{\text{dnn}}}$: Learned weights in (4).
Input: DS_E : Efficient nearest neighbors data structure.
Input: $\mathcal{C}$: set of available configurations.
Input: $\boldsymbol{c}$: start configuration.
Input: k_{nn} : Number of nearest neighbors to return.
Input: k_c : Number of nearest neighbor candidates.
Output: $\mathcal{C}_{k_{nn}}(\boldsymbol{c})$ configurations in $\mathcal{C}$ closest to $\boldsymbol{c}$ w.r.t. learned swept volume.

```
1: // Find neighbors from $\boldsymbol{c}$ w.r.t. fast, low-fidelity $\mathcal{D}^{\text{sv}}_{\text{we}}$ model. Total of $k_c$ returned.
2: // $\mathcal{C}_{k_c}$ is a set that contains $k_c$ neighbor candidates.
3: $NN_{\mathcal{D}^{\text{sv}}_{\text{we}}}(\mathcal{C}_{k_c}) = \text{DS}_E.\text{getNeighbors}(\boldsymbol{c}, k_c, \mathcal{D}^{\text{sv}}_{\text{we}}, \mathcal{C})$
4:
5: // Compute swept volume estimates for $k_c$ neighbors with high accuracy deep model.
6: $NN_{HNS}(\mathcal{C}_{k_c}) = NN_{\mathcal{D}^{\text{sv}}_{\text{dnn}}}(\{x | (x, d) \in NN_{\mathcal{D}^{\text{sv}}_{\text{we}}}(\mathcal{C}_{k_c})\})$
7:
8: // Find the $k_{nn}$ closest points w.r.t. trained $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ model.
9: // $\mathcal{C}_{k_{nn}}$ is a set that contains $k_{nn}$ neighbor candidates.
10: $\mathcal{C}_{k_{nn}} = \arg min^{k_{nn}}_{(x) \in C_{k_c}} NN_{HNS}(\mathcal{C}_{k_c})$
11: **return** $\mathcal{C}_{k_{nn}}$
```

The hierarchical combination of the metrics within neighbor selection has several benefits. First, it enables the use of any efficient nearest neighbor data structure. Second, it greatly reduces the number of $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ queries, which are slower than computing the weighted Euclidean distance. Finally, it employs metrics trained specifically for swept volume prediction.

4 Swept Volume Properties

DNNs are a universal approximator for any bounded continuous function [12]. In this section we formalize the proposition that swept volume is Lipschitz continuous and bounded along a continuous trajectory, justifying using DNNs as approxmators. The detailed proof is included in the supplementary materials for interested readers.

Proposition 1. $\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ *along a continuous trajectory between two configuration points* c_1, c_2 *in the C-space above is Lipschitz continuous, i.e.*

$$\|\mathcal{SV}(\boldsymbol{c}_1, \boldsymbol{c}_2) - \mathcal{SV}(\boldsymbol{c}_1 + \Delta\boldsymbol{c}_1, \boldsymbol{c}_2 + \Delta\boldsymbol{c}_2)\| \leq K\|\Delta\boldsymbol{c}_1 + \Delta\boldsymbol{c}_2\|. \tag{7}$$

5 Evaluations

We evaluate our method on a 15 DOF planar manipulator, a free-floating rigid body and a fixed-based Kuka manipulator. A $\mathcal{D}^{\text{sv}}_{\text{we}}$ model and a $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ model are trained to learn $\widetilde{\mathcal{SV}}$ for each robot. We compare PRMs and RRTs that use HNS to ones that use Euclidean distance, $\mathcal{D}_{\text{E}}$, and $\mathcal{D}^{\text{sv}}_{\text{we}}$, the most widely used distance metrics for sampling-based planners [17, 9].

We highlight important settings used in our evaluation below and leave details in the supplementary material. The three DNNs used to learn $\widetilde{\mathcal{SV}}$ for the robots share the same hyper-parameters and training dataset construction parameters. The dataset is composed of one hundred thousand training samples per robot. Additional ten thousand evaluation samples are generated in the same manner as the training samples but are unseen by the estimators. We use the PRM and RRT implemented in OMPL [23]. PRM with HNS identifies $k_{nn} = 5$ nearest neighbors among $k_c = 10$ candidates to connect to, while RRT with HNS finds $k_c = 5$ candidates in order to identify the nearest configuration in the tree. With a single metric, PRM simply uses $k_{nn} = 5$, and RRT finds the nearest configuration without pre-filtering. Figure 3 shows the starts and goals of the three robots evaluated in four environments. The Kuka manipulator is evaluated in two pick and place inspired tasks: `Retrieve` (Figure 3(e, f)) and `Shuffle` (Figure 3(g, h)) with complex environments. For the Kuka and the 15 DOF manipulators, the joint angles describe a configuration. The position and rotation quaternion describe a configuration of the free-floating rigid body.

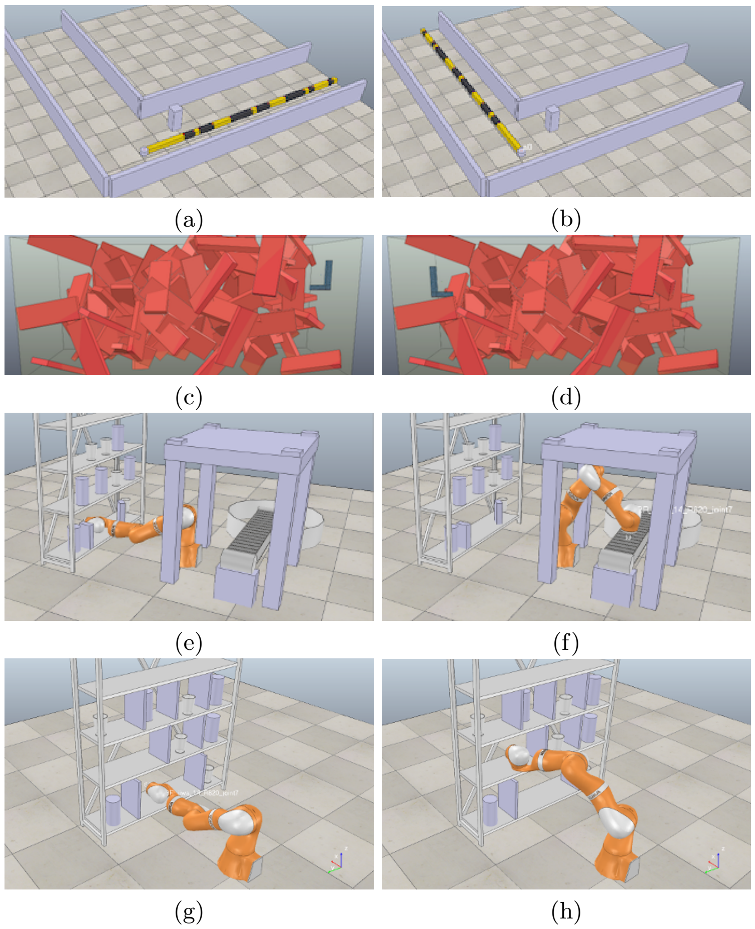
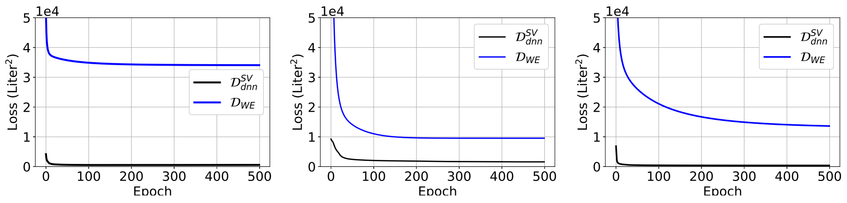


Fig. 3: *Start* (left column) and *goal* (right column) configurations of the 15 DOF planar manipulator (a, b), free-floating rigid body (c, d) and Kuka LBR IIWA 14 R820 manipulator in `Retrieve` task(e, f) `Shuffle` task (g, h).

5.1 Learning Results

The weights of $\mathcal{D}^{\text{sv}}_{\text{we}}$ for the 15 DOF and Kuka manipulators are approximately $[214, 97, 80, 73, 60, 43, 31, 28, 23, 19, 8, 5, 9, 7, 5]$ and $[1506, 2371, 181, 482, 1, 170, 30]$, respectively. As expected, these weights indicate that the joints near the base impact $\mathcal{SV}$ more. The weights of the free-floating rigid body are $[120, 140, 140]$ for x, y, z and $[86, 110, 82, 88]$ for quaternions. These weights indicate that the translational degrees of freedom has a higher impact on $\mathcal{SV}$ than rotation.

Figure 4 shows the evaluation loss of $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ and $\mathcal{D}^{\text{sv}}_{\text{we}}$ at each epoch. It is clear that learning converges for both $\mathcal{D}^{\text{sv}}_{\text{we}}$ and $\mathcal{D}^{\text{sv}}_{\text{dnn}}$, but $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ has a much smaller loss than $\mathcal{D}^{\text{sv}}_{\text{we}}$ across all robots. This means that $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ learns to approximate $\widetilde{\mathcal{SV}}$ much better than $\mathcal{D}^{\text{sv}}_{\text{we}}$.



(a) 15 DOF (b) Rigid Body (c) Kuka

Fig. 4: Learning curves of $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ and $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$.

Figures 5 shows the histogram of $\mathcal{D}_{\mathrm{E}}$, $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ and $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ for the evaluation data compared to the ground truth $\widetilde{\mathcal{SV}}$ (gray shade). Note that in order to compare to $\widetilde{\mathcal{SV}}$, we scale the value of $\mathcal{D}_{\mathrm{E}}$ such that the average matches the average $\widetilde{\mathcal{SV}}$ of the evaluation data. The last column of Figure 5 (c, f, i) shows striking similarities between $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ and $\widetilde{\mathcal{SV}}$, indicating the DNNs learned to approximate $\widetilde{\mathcal{SV}}$ well for all robots. In contrast, the highly nonlinear $\widetilde{\mathcal{SV}}$ of the 15 DOF (Figure 5 (a, b)) and Kuka manipulators (Figure 5 (g, h)) are not approximated well by $\mathcal{D}_{\mathrm{E}}$ and $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$. These linear metrics only approximate the free-floating rigid body well (Figure 5 (d, e)). This is expected as the joint angles in the articulated bodies nonlinearly impact each other. $\mathcal{D}_{\mathrm{E}}$ and $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ provide better approximations for the 7 DOF Kuka manipulator in 3D workspace than the 15 DOF planar manipulator. This is likely due to the fact that the Kuka manipulator has fewer DOF and similar lengths between joints. Similar trends can be found in Table 1, which shows the L1 norm of the error ratio for various distance metrics and robots w.r.t the evaluation dataset. The average error ratio of $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ is 4.38 to 11.67 times smaller than $\mathcal{D}_{\mathrm{E}}$ and 2.44 to 8.73 times smaller than $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$.

	15 DOF Manipulator	Rigid Body	Kuka Manipulator
$\mathcal{D}_{\mathrm{E}}$	76.9%	19.7%	60.7%
$\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$	70.7%	11.0%	29.9%
$\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$	**8.1%**	**4.5%**	**5.2%**

Table 1: L1 norm of error ratio $((\mathcal{D} - \widetilde{\mathcal{SV}})/\widetilde{\mathcal{SV}})$ for various distance metrics and robots. The metric with the lowest error ratio is highlighted for each robot.

We further explored the distance metric performance by comparing $\widetilde{\mathcal{SV}}$ against each distance metric (Figure 6). In this figure, the black squares, representing $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$, closely track $\widetilde{\mathcal{SV}}$. They are clustered along the diagonal for all robots. It is clear that neither $\mathcal{D}_{\mathrm{E}}$ (red circles) nor $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ (blue diamonds) correlate to $\widetilde{\mathcal{SV}}$ well for the 15 DOF or Kuka manipulator (Figure 6 (a, c)), especially when $\widetilde{\mathcal{SV}}$ is large or small.

We also investigated the learning performance of DNNs as impacted by the number of neurons in the hidden hidden layer and the quantity of training samples. Figure 7 shows the L2 loss over the evaluation dataset as a function of

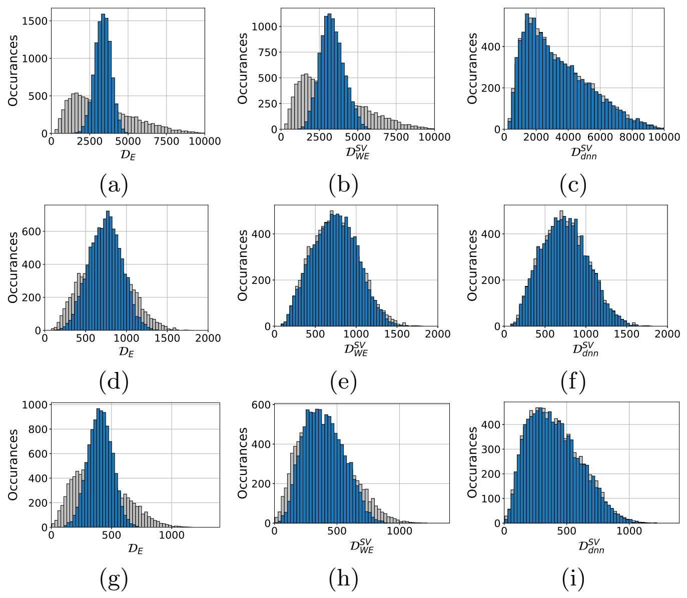
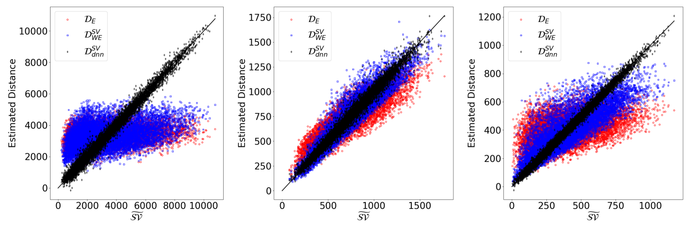


Fig. 5: Histogram of $\mathcal{D}_{\mathrm{E}}$ (left column), $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ (middle column) and $\mathcal{D}_{\mathrm{dnn}}^{\mathrm{sv}}$ (right column) for the 15 DOF manipulator (top row), free-floating rigid body (middle row) and Kuka manipulator (bottom row). The gray shade is the Histogram of $\widetilde{\mathcal{SV}}$.

training epochs as impacted by combinations of training sample and DNN sizes. It is clear that networks trained with 25,000 training samples (1/4 the original quantity of samples, shown as dashed lines) have higher loss across all robots and exhibit over-fitting as the loss increases after the initial decrease. When trained with the full training dataset (solid curves), large networks (networks with 1024 and 512 neurons in the first hidden layer) perform similarly across all robots while small networks demonstrate a larger loss for Kuka and rigid body robots. These results indicate that a large network and training dataset size are important to accurately approximate $\widetilde{\mathcal{SV}}$.

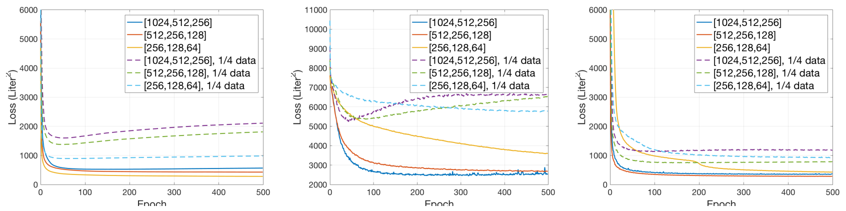
5.2 Planning Results

Next we compare the impact of various distance metrics on PRM and RRT. The top row of Figure 8 shows the cumulative success rate of identifying a collision-free motion plan as a function of time for PRM and RRT using the $\mathcal{D}_{\mathrm{E}}$ (red), $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ (blue) and HNS (black) distance metrics in various environments. For all scenarios with PRMs, using the HNS distance metric is more likely to successfully find a solution within the time budget in all cases. The gain in success rate at 200 s (max planning time allowed) compared to $\mathcal{D}_{\mathrm{E}}$ ranges from 1.27 to 5 times more while the gain over $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ ranges from 1.08 to 2.14 times more. In addition, the bottom row of Figure 8 shows that paths identified by HNS have a smaller swept volume. Comparing across robots, results demonstrate



(a) (b) (c)

Fig. 6: Scatter plots of $\widetilde{\mathcal{SV}}$ and the distance estimated by $\mathcal{D}_{\mathrm{E}}$ (red circles), $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ (blue squares) and the DNN ($\mathcal{D}_{\mathrm{dnn}}^{\mathrm{sv}}$) (black diamonds) for the 15 DOF manipulator (a), free-floating rigid body (b) and Kuka manipulator (c).



(a) High DOF (b) Rigid body (c) Kuka

Fig. 7: The L2 evaluation loss of DNNs with varied numbers of neurons in the hidden layer as shown in the legend (solid curves) across robots. The dashed curves show the same network trained with 25,000 training samples (1/4 of the full sample size).

that the advantage of HNS is much less prominent for the rigid body robot. This is expected since $\mathcal{D}_{\mathrm{E}}$ and $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ both approximate $\widetilde{\mathcal{SV}}$ reasonably well for this system. HNS shows a similar performance gain for both the 3D Kuka and the 2D 15 DOF manipulators. In the RRT case, HNS also enhances planning by identifying solutions with lower swept volume and identifying solutions where other metrics failed. For example, Figure 8 (a) clearly shows that the 15 DOF manipulator in a narrow corridor is very difficult for RRT as neither $\mathcal{D}_{\mathrm{E}}$ nor $\mathcal{D}_{\mathrm{we}}^{\mathrm{sv}}$ found a solution in 20 runs. In contrast, RRTs using HNS were able to identify a solution in 2 runs, likely due to the goal bias mechanism of RRT which has been shown to significantly increase the performance of RRT [11]. In the planning scenario shown in Figure 1 (a, b), the start and goal have the same joint angles except for the joint at the base. This means $\mathcal{D}_{\mathrm{E}}$ between the start and goal is relatively small. However, the robot must curl towards the base and then extend in order to reach the goal. These curled configurations require a large $\mathcal{D}_{\mathrm{E}}$ change from the goal configuration and therefore are unlikely to be selected by an RRT using goal bias. As a result, the goal bias is ineffective for the Euclidean-based metrics as it mostly selects configurations near the start. In contrast, HNS does not have this problem since the $\mathcal{SV}$ between the start and goal is larger than the $\mathcal{SV}$ between any curled configuration and the goal.

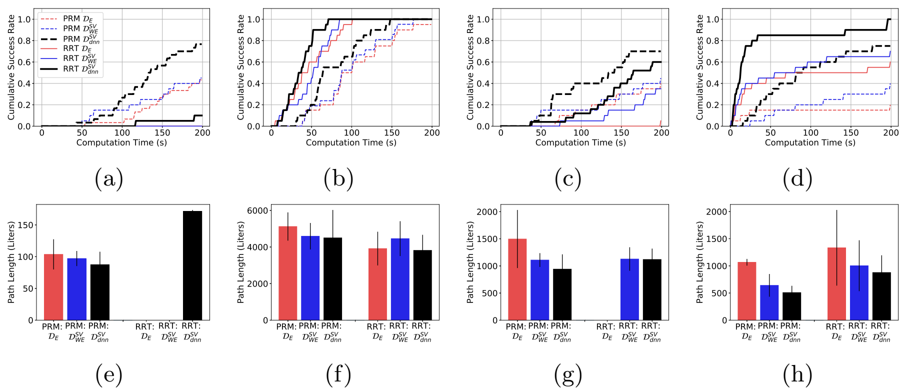


Fig. 8: The cumulative success rate of identifying a path (top row) and path length (in units of $\widetilde{\mathcal{SV}}$) of successful runs (bottom row) for PRM (dotted lines) and RRT (solid lines) evaluated on the 15 DOF manipulator (a, e), free-floating rigid body (b, f), Kuka manipulator in the `Retrieve` task (c, g) and in the `Shuffle` task (d, h). The color of bars and curves represents various distance metrics (red: $\mathcal{D}_{\mathrm{E}}$, blue: $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$, black: $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ (our method)). The path length data is not available for RRTs using $\mathcal{D}_{\mathrm{E}}$ or $\mathcal{D}_{\mathrm{WE}}$ since there were zero successful runs.

6 Distance Metric Trade-Offs

In Section 5, the advantages of HNS are clear, particularly when $\mathcal{D}_{\mathrm{E}}$ or $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ cannot capture $\widetilde{\mathcal{SV}}$ well, i.e., when the robot has a highly articulated body. Here we investigate the advantages further by empirically evaluating the computational cost and the quality of returned nearest neighbors for each distance metric.

	Training			Distance Call		
Robot	Data Generation	$\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ Training	$\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ Training	Compute $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$	Compute $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$	Compute $\widetilde{\mathcal{SV}}$
15 DOF Manipulator	31hr	630.02s	4360.03s	0.081μs	175.1μs	8.85s
Free-floating Rigid Body	2hr	601.53s	4001.53s	0.053μs	164.3μs	0.58s
Kuka Manipulator	14hr	629.33s	4023.35s	0.055μs	164.3μs	4.06s

Table 2: Computation time of various operations broken down by training (Training) and a single usage as done as a primitive operation in motion planning (Distance Call).

The computation time required by the distance metrics is shown in Table 2. Recall that $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ and $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ are trained once per robot (columns 3 and 4) and utilized one hundred thousand training samples (column 2). After training, the computation time for a single inference to $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ is at or just under 175μs (column 6). Comparing a single inference to times required to generate $\widetilde{\mathcal{SV}}$ for each robot shows that $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ inference is 3500 to 5000 times faster than state of the art $\widetilde{\mathcal{SV}}$ computation. In addition, the computation time of $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$ is only slightly affected by the DOF of the robot and is independent to the robot's 3D model complexity. On the other hand, computing $\mathcal{D}^{\mathrm{sv}}_{\mathrm{we}}$ (column 5) is about 2000 to 3000 times faster than querying $\mathcal{D}^{\mathrm{sv}}_{\mathrm{dnn}}$. These results suggest HNS can reduce computation time as it identifies candidate nearest neighbors using the fast $\mathcal{D}^{\text{sv}}_{\text{we}}$ before the slower, more accurate, $\mathcal{D}^{\text{sv}}_{\text{dnn}}$.

	Percent of Non-Matching Neighbors				Percent Additional Volume Swept			
Robot	$\mathcal{D}_{\text{E}}$	$\mathcal{D}^{\text{sv}}_{\text{we}}$	$\mathcal{D}^{\text{sv}}_{\text{dnn}}$	HNS	$\mathcal{D}_{\text{E}}$	$\mathcal{D}^{\text{sv}}_{\text{we}}$	$\mathcal{D}^{\text{sv}}_{\text{dnn}}$	HNS
15 DOF	87%	80%	**32%**	65%	180%	160%	**14%**	61%
Rigid body	65%	22%	**11%**	**11%**	38%	7%	**2%**	**2%**
Kuka	87%	33%	**12%**	15%	56%	8%	**1%**	2%

Table 3: Comparison of neighboring configurations selected by various distance metrics as compared to those selected by $\widetilde{\mathcal{SV}}$. The Percent of Non-Matching Neighbors columns demonstrate the quantity of neighboring configurations that do not match those selected by $\widetilde{\mathcal{SV}}$. The Additional Swept Volume columns capture the amount of additional volume swept by the neighboring configurations selected by the metrics as over that of the $\widetilde{\mathcal{SV}}$ configurations. The best metric of each robot is highlighted.

It is clear from Section 5 that nearest neighboring configurations selected w.r.t. $\mathcal{D}_{\text{E}}$ and $\mathcal{D}^{\text{sv}}_{\text{we}}$ are very different from ones selected by HNS. However, little is known about the quality of neighboring configurations returned by the distance metrics. We evaluate this by comparing neighbors returned by each metric to those returned by $\widetilde{\mathcal{SV}}$ comparison. Since a full comparison during a planning run would be computationally prohibitive, we randomly sample 100 starting configurations ($\boldsymbol{c}_1$) and 100 potential neighbor configurations ($\boldsymbol{c}_2$) for each robot. For each $\boldsymbol{c}_1$, five nearest configurations among $\boldsymbol{c}_2$ are identified using $\mathcal{D}_{\text{E}}$, $\mathcal{D}^{\text{sv}}_{\text{we}}$, $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ and HNS ($k_c = 10$). Then, these configurations are compared to the configurations selected by $\widetilde{\mathcal{SV}}$. Table 3 captures the quantity and quality differences in the returned neighbor configurations. First, the percentage of configurations returned by each metric that do not match those returned by $\widetilde{\mathcal{SV}}$ are shown. Next, the quality of the returned configurations for each metric is demonstrated by tallying the additional volume swept by the returned neighbors over the baseline provided by the configurations returned by $\widetilde{\mathcal{SV}}$. These values demonstrate that $\mathcal{D}_{\text{E}}$ selects very different neighboring configurations than $\widetilde{\mathcal{SV}}$, in one example incurring a 171% increase in swept volume. In contrast, $\mathcal{D}^{\text{sv}}_{\text{we}}$, with weights optimized to mimic $\widetilde{\mathcal{SV}}$, chooses more similar neighboring configurations with only an up to 7% increase in additional volume swept for the L-shaped and Kuka manipulator robot. However, the simple weights face difficulty capturing the highly nonlinear $\mathcal{SV}$ of the 15 DOF manipulator well, resulting in a 166% increase in volume swept. In contrast, HNS chooses neighboring configurations closest to $\widetilde{\mathcal{SV}}$, and the additional volume swept is much lower than any other Euclidean-based metric, i.e., 2.7 to 3.5 times smaller than $\mathcal{D}^{\text{sv}}_{\text{we}}$. As expected, $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ selects neighboring configurations closest to $\widetilde{\mathcal{SV}}$ for all robots tested. However, computing $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ is much slower than HNS, and efficient nearest neighbor data structures cannot be used since $\mathcal{D}^{\text{sv}}_{\text{dnn}}$ does not form a metric space.

7 Conclusion

The ability of DNNs to approximate any continuous bounded function makes them especially well suited to estimate swept volume. We demonstrated this ability for several multibody systems, from a rigid body to manipulator systems. To further enhance efficiency for use in complex sampling-based motion planning scenarios, we integrated the DNN with a trained weighted Euclidean metric. The hierarchical combination of learned metrics retains metric space properties along with high-fidelity. This hierarchical combination of metrics improved the performance of both RRT and PRM planners in all scenarios tested, particularly when the robot has a highly articulated body.

8 Acknowledgement

Tapia, Chiang, and Sugaya are partially supported by the National Science Foundation under Grant Numbers IIS-1528047 and IIS-1553266 (Tapia, CAREER). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Abrams, S., Allen, P.K.: Computing swept volumes. The Journal of Visualization and Computer Animation **11**(2) (2000) 69–82
2. Abdel-Malek, K., Yang, J., Blackmore, D., Joy, K.: Swept volumes: foundation, perspectives, and applications. International Journal of Shape Modeling **12**(01) (2006) 87–127
3. Himmelstein, J.C., Ferre, E., Laumond, J.P.: Swept volume approximation of polygon soups. IEEE Trans. on Autom. Sci. and Eng. **7**(1) (2010) 177–183
4. Kuffner, J.J.: Effective sampling and distance metrics for 3D rigid body path planning. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2004) 3993–3998
5. Kim, Y.J., Varadhan, G., Lin, M.C., Manocha, D.: Fast swept volume approximation of complex polyhedral models. Computer-Aided Design **36**(11) (2004) 1013–1027
6. Von Dziegielewski, A., Hemmer, M., Schömer, E.: High precision conservative surface mesh generation for swept volumes. IEEE Trans. on Autom. Sci. and Eng. **12**(1) (2015) 183–191
7. Campen, M., Kobbelt, L.: Polygonal boundary evaluation of minkowski sums and swept volumes. In: Computer Graphics Forum. Volume 29. (2010) 1613–1622
8. Ekenna, C., Uwacu, D., Thomas, S., Amato, N.M.: Improved roadmap connection via local learning for sampling based planners. In: Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS). (2015) 3227–3234
9. Amato, N.M., Bayazit, O.B., Dale, L.K., Jones, C., Vallejo, D.: Choosing good distance metrics and local planners for probabilistic roadmap methods. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (1998) 630–637
10. Kavraki, L., Svestka, P., claude Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (1996) 566–580

11. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5) (2001) 378–400
12. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks **4**(2) (1991) 251–257
13. Brin, S.: Near neighbor search in large metric spaces. In: Proc. of Int. Conf. on Very Large Data Bases. (1995) 574–584
14. Perrin, N., Stasse, O., Baudouin, L., Lamiraux, F., Yoshida, E.: Fast humanoid robot collision-free footstep planning using swept volume approximations. IEEE Trans. Robot. **28**(2) (2012) 427–439
15. Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: A review. IEEE Access **2** (2014) 56–77
16. Voelz, A., Graichen, K.: Distance metrics for path planning with dynamic roadmaps. In: Proc. Int. Symp. on Robotics. (2016) 126–132
17. Palmieri, L., Arras, K.O.: Distance metric learning for RRT-based motion planning with constant-time inference. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2015) 637–643
18. Wolfslag, W.J., Bharatheesha, M., Moerland, T.M., Wisse, M.: RRT-CoLearn: towards kinodynamic planning without numerical trajectory optimization. IEEE Robot. Autom. Lett. **3**(3) (2018) 1655–1662
19. Faust, A., Ramirez, O., Fiser, M., Oslund, K., Francis, A., Davidson, J., Tapia, L.: PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2018) In press
20. http://www.cs.unm.edu/tapialab/Resources/SweptVolume/index.php
21. Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789) (2000) 947
22. McMahon, T., Jacobs, S., Boyd, B., Tapia, L., Amato, N.M.: Local randomization in neighbor selection improves PRM roadmap quality. In: Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS). (2012) 4441–4448
23. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robot. Automat. Mag. **19**(4) (December 2012) 72–82 http://ompl.kavrakilab.org.