2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Deep Prediction of Swept Volume Geometries: Robots and Resolutions

John Baxter[1], Mohammad R. Yousefi[1], Satomi Sugaya[1], Marco Morales[2] and Lydia Tapia[1]

***Abstract*— Computation of the volume of space required for a robot to execute a sweeping motion from a start to a goal has long been identified as a critical primitive operation in both task and motion planning. However, swept volume computation is particularly challenging for multi-link robots with geometric complexity, e.g., manipulators, due to the non-linear geometry. While earlier work has shown that deep neural networks can approximate the swept volume quantity, a useful parameter in sampling-based planning, general network structures do not lend themselves to outputting geometries. In this paper we train and evaluate the learning of a deep neural network that predicts the swept volume geometry from pairs of robot configurations and outputs discretized voxel grids. We perform this training on a variety of robots from 6 to 16 degrees of freedom. We show that most errors in the prediction of the geometry lie within a distance of 3 voxels from the surface of the true geometry and it is possible to adjust the rates of different error types using a heuristic approach. We also show it is possible to train these networks at varying resolutions by training networks with up to 4x smaller grid resolution with errors remaining close to the boundary of the true swept volume geometry surface.**

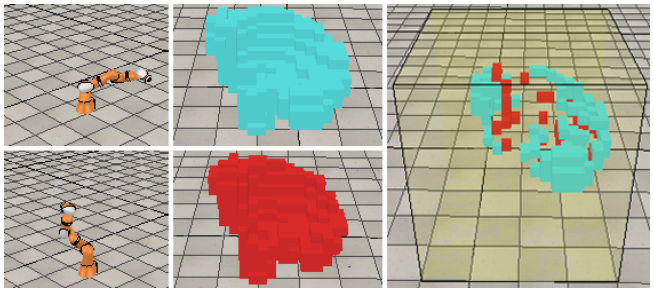

Fig. 1. Illustration of swept volume geometry for a linear joint interpolation of a 7 degree of freedom Kuka robot model. Start (left, top) and end (left, bottom) joint positions are recorded as configurations $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, respectively. The recorded swept volume labels (middle, top in blue) are used for training and evaluation. The prediction (middle, bottom in red) from the network is used to show errors (right). False positive errors are shown in red and false-negative errors are shown in blue. The yellow-box in the error image shows the bounding box (V) from Table I.

## I. Introduction

The swept volume between two robot configurations is the physical volume that the robot occupies during its motion from one configuration to the other (Fig. 1). Since some of the earliest work in motion planning, its computation has been recognized as a critical primitive operation in planning [1]. This is because the swept volume, $\mathcal{SV}(\boldsymbol{c}_0, \boldsymbol{c}_1)$, between two robot configurations, $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, can be used for collision detection. While the computation of this swept volume is direct for simple rigid bodies, the computation can quickly become intractable for objects with rotations and for objects that are geometrically complex [2]. This was noted even in this earliest work [1], where the problem being considered was industrial manipulation applications. More recently, swept volume computation is still a critical element of modern sampling-based motion planning, for example for continuous collision detection and planning in high dimensional spaces [3], [4], [5]. Other applications include collision detection for industrial and humanoid robots [6] and foot-step planning for legged robots [7]. The scalar size of the swept volume has also been used as a distance metric [8] and as a measure of compactness of the workspace [9]. However, due to computational cost, approximations are often employed [10], [11]. Moreover, the use of the swept volume has been recognized in the efficient linking between motion and task planning [12], [13].

The computation of swept volumes for single rigid bodies along simple paths can be performed efficiently [14], [15], however exact swept volumes for articulated bodies are in general intractable by analytic and algebraic methods [14], [16] due to the complex non-linear geometry. Approximation of swept volume has thus been a focus of many algorithms, such as convex polyhedra-based, occupation grid-based, and boundary-based methods [17], [18], [6], [12], [19], [15], [20]. These algorithms can still not be sufficient for some motion planning applications due to high computational expense and overly conservative estimates [12], [10]. We recently demonstrated that the general approximation ability of deep neural networks facilitates the approximation of the scalar value of swept volumes, $|\mathcal{SV}|$, that can be directly used in sampling-based planning for filtering local planning attempts [8]. However, predicting a geometric representation of the actual volume swept would require another approach. In a single proof of concept, a network was used to output a geometric volume of swept volume for a single robot [21]. Here, we report an extensive evaluation of learned swept volume geometry for multiple robot types and we explore

* Baxter, Yousefi, and Sugaya contributed equally. This work is partially supported by the National Science Foundation (NSF) under Grant Numbers IIS-1528047 and IIS-1553266, by the Air Force Research Laboratory (AFRL) under agreement number FA9453-18-2-0022, and by Asociación Mexicana de Cultura A.C. Any opinions, findings, conclusions, recommendations, or views contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, AFRL or U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

[1]Department of Computer Science, University of New Mexico, MSC01 11301 University of New Mexico, Albuquerque, NM 87131, USA automata@unm.edu, myousefi@unm.edu, satomi@cs.unm.edu, and (corresponding) tapia@cs.unm.edu

[2]Department of Digital Systems, Instituto Tecnológico Autónomo de México, Rio Hondo 1, Col. Progreso Tizapán, C.P. 01080 México marco.morales@itam.mx

the impact of the resolution on the prediction.

This work offers several contributions. First, we show it is possible to approximate the geometry of the swept volume using deep neural networks (DNNs) for a wide variety of robots. Second, despite the inability of networks to guarantee error bounds on predictions we show empirically that the errors are largely limited to areas near the boundary and the network output can be tuned to prioritize one type of error over another. Specifically, we use the heuristic parameter $\tau$ to adjust the error for conservative or aggressive prediction of the swept volume geometry. Third, we analyze the effect of finer resolution on network learning and show that the network errors are not affected by the change in resolution allowing for better approximations of the swept volume geometry by increasing the sampling resolution in the volume space. Finally, in support of these contributions and wide use of this work, we have publicly made available the data and networks used in this study (http://tapialab.science/Resources). Visualizations of example predictions are provided in the attached video.

## II. Related Work

As the computation of swept volume becomes intractable for objects with rotations and complex geometries, there are roughly three classes of modern swept volume approximation algorithms: boundary-based, convex polyhedra-based and occupation grid-based methods. In boundary-based methods an outer boundary surface of the swept volume is computed. This is done by developing surfaces for sweeping and rotational motion of the object surfaces and computing the surface of the union [20], [17], [15]. These methods focus on computing swept volume boundaries of moving solids and are generally slow, especially for articulated robots. In convex polyhedra-based methods, robot bodies are approximated with convex hulls [22]. These methods generally decompose the model into small sets of convex polyhedrons which itself is a challenging problem. When the robot moves between configurations, additional convex hulls are added and the union of convex hulls is the swept volume [12]. In occupancy grid-based methods, the space is split into voxels. These methods then record which segments of space are touched by the robot as it moves in steps between the configurations [18], [19]. In our prior work, we trained a DNN to predict scalar swept volumes after training with a set of scalar volume data derived from swept volume occupancy grids [8]. In contrast, this work provides a network structure to directly predict an occupancy grid.

Deep neural networks have been used to generate 3D volumetric models from 2D images [23], [24]. In recent work, multiple images from unknown perspectives were used to predict the shape of an object [23]. Even more recently, an alternative network design demonstrated that it could generate the 3D shape model from a single image [24]. Additionally, neural networks have been used to fill the missing volume in an incomplete 3D volume of an object [25]. Similar to these works, we output 2D or 3D volumetric models from deep neural networks. However, our inputs are representations of the two robot configurations, *e.g.*, sets of angles representing joint positions.

## III. Swept Volume

The true swept volume geometry, $\mathcal{SV}$, for a straight line configuration space trajectory can be formulated as the union of workspace robot poses corresponding to the trajectory's configurations [8],

$$\mathcal{SV}\left(\boldsymbol{c}_0, \boldsymbol{c}_1\right) = \cup_{t\in[0,1]}\mathcal{V}((1-t)\boldsymbol{c}_0 + t\boldsymbol{c}_1), \tag{1}$$

where $\boldsymbol{c}_0$, $\boldsymbol{c}_1$ are start and end robot configuration vectors in $\mathbb{R}^d$ and $d$ is the number of Degrees Of Freedom (DOF) of the robot. $\mathcal{V}$ maps configurations to workspace geometry representing the volume swept between $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, and $t$ is the parameterized time, $t \in [0, 1]$. As elaborated in Sec. II, the exact calculation of $\mathcal{SV}$ is often intractable, therefore we compute the approximate swept volume geometry function, $\mathcal{SV}_G$.

The learning problem for the scalar-function counterpart of $\mathcal{SV}_G$, *i.e.*, the function that maps robot configuration pairs to corresponding swept volume scalar values, has been formulated as a continuous function approximation via DNNs and the learning has been shown to be successful [8]. This function approximation was formulated on the basis of the Lipschitz continuity of the scalar function for articulated bodies [21]. Although the output of the swept volume *geometry* function is continuous in principle, for practical reasons it is represented as a discretized 3D space. Moreover, the training data would also be generated in a discretized space. Therefore, we formulate the computation of $\mathcal{SV}_G$ as a classification problem as follows.

Given a discretized 3D volume of fixed size, $V$, let $\boldsymbol{r}_{i,j,k}$ denote the coordinate of a grid point along $x$, $y$, and $z$ Cartesian coordinates. The subscripts $i$, $j$, $k$, denotes locations along $x$, $y$, $z$, respectively, and are bounded by the number of voxels along each dimension (fixed for each robot). We formulate the swept volume geometry function we aim to learn, $\mathcal{SV}_G(\boldsymbol{c}_0, \boldsymbol{c}_1)$, as a function whose input is a configuration pair, $\boldsymbol{c}_0$, $\boldsymbol{c}_1$, and whose output is the occupation probability of a voxel centered at $\boldsymbol{r}_{i,j,k}$, $P(\boldsymbol{r}_{i,j,k})$, $\forall\, i, j, k \in V$. The occupation probability is then classified into two classes: occupied (on) or empty (off). The threshold of the probability that separates these classes is an adjustable parameter, $\tau$.

The learned function and the predicted occupation probability are denoted with tilde: $\widetilde{\mathcal{SV}}_G$ and $\widetilde{P}(\boldsymbol{r}_{i,j,k})$. We define the cost function parameterized by $\theta$ as

$$J_\theta = \frac{1}{N_V}\sum_{i,j,k\in \mathrm{V}}\Big(P(\boldsymbol{r}_{i,j,k}) - \widetilde{P}(\boldsymbol{r}_{i,j,k})\Big)^2, \tag{2}$$

where $N_\mathrm{v}$ is the total number of voxels in V. The learning objective is to minimize this function with respect to $\theta$. This cost function was the one that produced the best results as explained in Sec. IV-C.

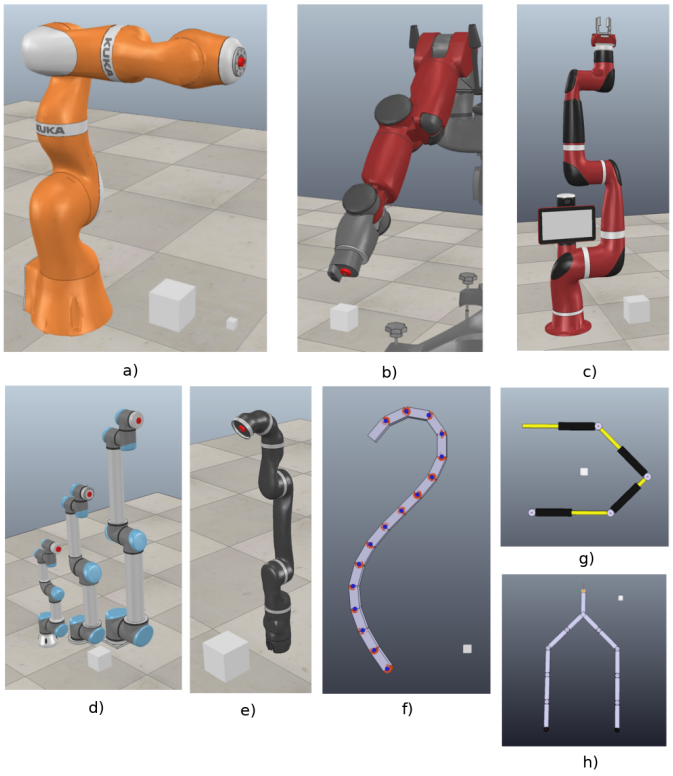


Fig. 2. Robots (and cubic voxel of width 0.1 m, for reference): a) Kuka manipulator ($\text{Kuka}_{20}$) with an additional voxel of width 0.025 m (for high-resolution $SV_G$ reference), b) Baxter arm (Baxter), c) Sawyer manipulator (Sawyer), d) (left to right) UR3, UR5, and UR10, e) Mico manipulator (Mico), f) 2D planar manipulator (Planar), g) 2D prismatic-revolute manipulator (PR), and h) 2D closed loop manipulator (CL).

| Robot | DOF | V ($x \times y \times z$) | Voxel Width (m) | Space Usage (%) |
|---|---|---|---|---|
| $\text{Kuka}_{10}$ †‡ | 7 | $10 \times 10 \times 10$ | 0.1 | 67.0 |
| $\text{Kuka}_{15}$‡ | 7 | $15 \times 15 \times 15$ | 0.1 | 59.2 |
| $\text{Kuka}_{20}$‡ | 7 | $20 \times 20 \times 20$ | 0.1 | 50.5 |
| $\text{Kuka}_{25}$‡ | 7 | $25 \times 25 \times 25$ | 0.1 | 51.9 |
| $\text{Kuka}_{30}$‡ | 7 | $30 \times 30 \times 30$ | 0.1 | 48.5 |
| $\text{Kuka}_{80}$‡ | 7 | $80 \times 80 \times 80$ | 0.025 | 42.4 |
| Baxter† | 7 | $22 \times 22 \times 21$ | 0.1 | 53.1 |
| Sawyer† | 8 | $26 \times 26 \times 26$ | 0.1 | 39.5 |
| UR3† | 6 | $15 \times 15 \times 15$ | 0.1 | 42.3 |
| UR5† | 6 | $20 \times 20 \times 20$ | 0.1 | 59.9 |
| UR10† | 6 | $30 \times 30 \times 30$ | 0.1 | 46.8 |
| Mico† | 6 | $15 \times 15 \times 15$ | 0.1 | 35.1 |
| Planar† | 16 | $1 \times 100 \times 100$ | 0.1 | 42.3 |
| PR† | 8 | $88 \times 88 \times 2$ | 0.1 | 68.5 |
| CL† | 9 | $41 \times 50 \times 3$ | 0.1 | 93.8 |

TABLE I

ROBOTS FROM THE BASE STUDY(†) AND RESOLUTION STUDY(‡) USED IN THIS WORK. ROBOT DOF, SIZE OF BOUNDING VOLUME (V), AND WIDTHS OF A CUBIC VOXEL (VOXEL WIDTHS). SPACE USAGE IS A MEASURE OF THE UNION OF THE VOLUMES SAMPLED IN THE TRAINING DATA OVER THE FULL BOUNDING VOLUME.

## IV. METHODS

### A. Robot Models

We consider swept volume predictors for the fixed-base robot models shown in Fig. 2. These robots have a variety of joint properties and DOFs. We refer to these robots as the *base robot set*. The motions of Planar, PR, and CL are confined to 2D while all other robots move in 3D.

### B. Training Dataset Generation

Since the union in Eq. (1) is difficult and expensive to compute exactly, we approximate it through a discretized grid of fixed resolution. We use the following method to generate training and evaluation data.

For simplicity, the limits of the grid in each of the three dimensions define a cuboid that subsumes the generated volume. The dimension of this cuboid, V, is fixed for each robot (see Table I) and includes all reachable and some unreachable space for the robot. On-voxels are given a value of 1 while off-voxels are given a value of 0.

The input features of the data, configuration pairs $(\boldsymbol{c}_0, \boldsymbol{c}_1)$ are sampled uniformly at random. We implement Eq. 1 in V-REP by discretizing the range of $t$ using $\Delta t$ step size to form a configuration space straight line interpolation generating the corresponding swept volume geometry. In each step of the interpolation, the volume of the robot body corresponding to the configuration is added to an octree, similar to [19]. The label data for this volume is in the form of a voxel grid created from the octree at $t = 1$. All swept volume generation was done using the built in function sim.insertObjectIntoOctree from V-REP [26] on a machine with an Intel Xeon E-2146G (@3.5 GHz) processor with 6 cores and an Nvidia Quadro P1000 accelerator.

It should be noted that sampling configuration pairs for CL was done in consideration of its end effector. First, pairs of end-effector poses are selected at random from the reachable space of the end effector. Then, the corresponding joint positions are determined through inverse kinematics. Next, each interpolation step moves the end effector of the robot linearly between beginning and final end-effector poses.

Additionally, in order to investigate the effect of training data resolution on the quality of $\widetilde{\mathcal{SV}}_G$, we produced various scaled-size models of Kuka and generated data while keeping fixed the voxel size. Smaller to larger Kuka models correspond to lower to higher data resolution. Five levels of resolutions are considered in this way and we denote each dataset as $\text{Kuka}_{10}$, $\text{Kuka}_{15}$, $\text{Kuka}_{20}$, $\text{Kuka}_{25}$, and $\text{Kuka}_{30}$. The subscripts indicate the size of the cuboid bounding volume, V. For example $\text{Kuka}_{10}$ has V=$10\times10\times10$ voxels. One higher resolution was considered, $\text{Kuka}_{80}$, for which the voxel size was scaled down to 0.015625 liters (a cube of width 0.025 m) while keeping the robot size the same as $\text{Kuka}_{20}$. We refer to these Kuka models/data as the *resolution study set*.

### C. Network Architectures, Swept Volume Geometry Function Training, and Prediction

We train swept volume geometry function approximators, $\widetilde{\mathcal{SV}}_G$, in a supervised manner. Two DNN architectures are used to represent $\widetilde{\mathcal{SV}}_G$; one for $\text{Kuka}_{80}$ and the other for all

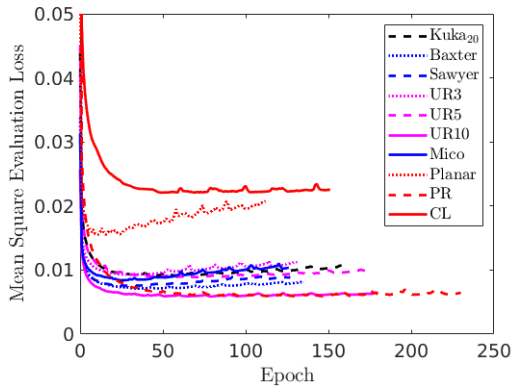


Fig. 3. Evaluation loss as a function of training epochs. Convergence criterion used is 100 epochs with no improvement in lowest evaluation loss. Some curves demonstrate over-fitting (e.g., Planar), therefore the parameters with the lowest evaluation loss observed was used in those cases.

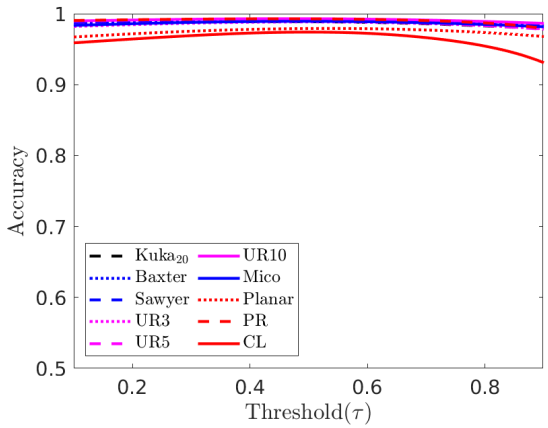


Fig. 4. Prediction accuracy as a function of output probability threshold ($\tau$) used to classify on/off voxels for base robot set.

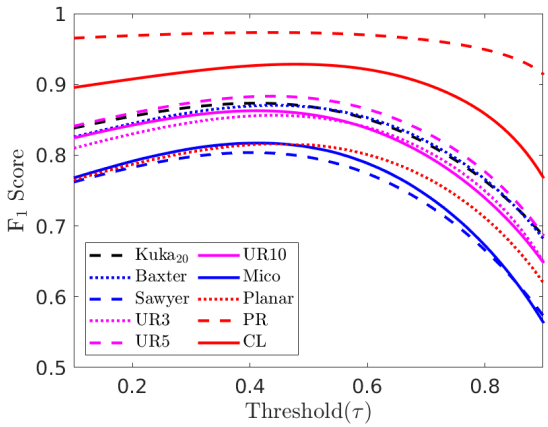


Fig. 5. Prediction $F_1$ Score as a function of output probability threshold ($\tau$) used to classify on/off voxels for base robot set.

other robots. This is due to a large output dimension required for $\text{Kuka}_{80}$. Specifically, creating a fully connected layer with 512,000 neurons as the output layer requiring 4000 $\times$ 512,000 connections in our model is prohibitively expensive.

First, we describe the DNN architecture representing all robots' $\widetilde{\mathcal{SV}}_G$ except for $\text{Kuka}_{80}$. We draw inspiration from decoder networks [27] since the output dimensions are far larger than the input dimensions. Decoder network structures can construct a large space swept by the robot in motion from a set of small but expressive features, *i.e.*, joint configurations. This is a simple fully connected feed-forward network of four hidden layers with [500, 1000, 2000, 4000] ReLu neurons. Input and output layer dimensions depend on the dimensions of each robot's input features (DOF), $(\boldsymbol{c}_0, \boldsymbol{c}_1)$, and output volume, V (see Table I). The output layer also has ReLu neurons.

The network representing $\widetilde{\mathcal{SV}}_G$ for $\text{Kuka}_{80}$ starts with a similar decoder network as the other robots followed by three deconvolution layers [28]. The decoder network has three hidden layers and one output layer with [125, 250, 500] and 1000 ReLu neurons, respectively. This output layer is fed to three successive deconvolution layers each with 16 filters, 3×3×3 kernels, and stride of 2×2×2. The output is an 80×80×80 voxel grid.

These networks were trained using the cost function given in Eq. (2) via stochastic gradient descent using the Adam optimizer [29] with learning rate 0.0001, beta1 0.9, beta2 0.999, and epsilon $1e^{-8}$. For each robot, 90,000 training samples and 10,000 evaluation samples (with no overlap) were used with a mini-batch size of 100. Although we initially tried a cross-entropy loss since it is common in classification (*e.g.*, [30]), the learning did not converge. However, the mean square loss (MSE), as in Eq. (2), provided better performance.

The learning curves show stable learning with over-fitting trends, with some robots demonstrating more over-fitting than others (Fig. 3). Network training ends at the end of the 100th training epoch without improvement in the evaluation loss and the DNN at the lowest evaluation loss is chosen as the $\widetilde{\mathcal{SV}}_G$ for each robot.

To train each network, we used an Intel Xeon Gold 6148 Processor with 20 cores @ 3.7 GHz. The network training is accelerated by a single Nvidia Tesla V100 GPU. Please note that this system is different from the training swept volume generation as TensorFlow is optimized for GPU performance.

A trained $\widetilde{\mathcal{SV}}_G$ outputs occupation probability of each voxel centered at $\boldsymbol{r}_{i,j,k}$, $\widetilde{P}(\boldsymbol{r}_{i,j,k})$, for all $i, j, k \in$ V. Each voxel is then classified as on or off using the threshold $\tau \in [0, 1]$; on when $\widetilde{P}(\boldsymbol{r}_{i,j,k}) \geq \tau$ and off when $\widetilde{P}(\boldsymbol{r}_{i,j,k}) < \tau$. All the on voxels together represents the approximated swept volume geometry.

## V. Evaluation

### A. $\widetilde{\mathcal{SV}}_G$ Prediction Quality

We evaluate the quality of the learned swept volume geometry function ,$\widetilde{\mathcal{SV}}_G$, using two metrics, accuracy and $F_1$ score. *Accuracy* is computed as the ratio of correctly-predicted voxel count to the total voxel count. $F_1$ *score* is

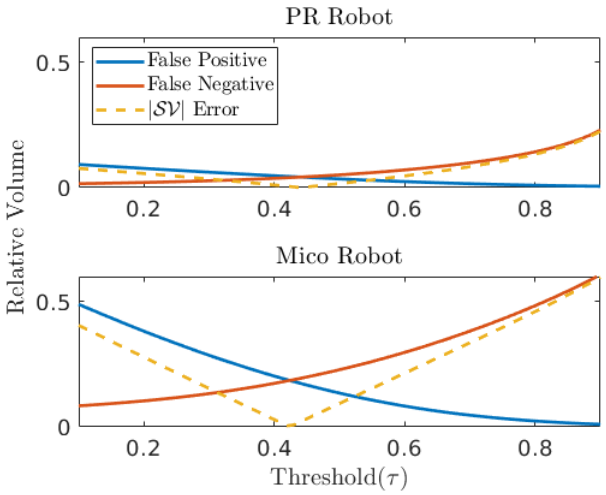


Fig. 6. False-positive (blue) and false-negative (red) voxel counts as a function of output probability threshold ($\tau$) used to classify voxels for PR (top) and Mico (bottom) Robots. Error in the scalar value of swept volume, $|\mathcal{SV}|$, is also plotted (yellow dotted).

the harmonic mean of the precision and recall, *i.e.*,

$$\mathrm{F}_1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{3}$$

Precision is computed as the ratio of true-positive prediction to predicted volume, and recall is computed as the ratio of true-positive prediction to true-volume. $\mathrm{F}_1$ score of 1 indicates perfect geometry prediction.

$\widetilde{\mathcal{SV}}_G$ predicts the occupation value of the output volume at each grid location accurately for a wide range of threshold values, $\tau$, (Fig. 4), with lowest accuracy at 93.1% for the CL at $\tau = 0.9$ and highest value of 99.0% for UR5 at $\tau = 0.48$. We emphasize that accuracy includes correct prediction of both *on and off* voxels. (This is why accuracy values don't reach zero at low and high $\tau$ values.) One reason for high accuracy is the presence of always-off and always-on voxels within the bounding volume V. Always-on voxels would be the voxels near and at the robot base and always-off voxels belong to the unreachable space within V. For example, the Planar robot's reachable space would be a disk whose radius is the length of its kinematic chain. Thus, the unreachable space is the remainder of the space within V. A simple calculation using basic geometry of disks and spheres yield that the percentage of always-off volume is about 21% for 2D robots and 48% for 3D for a circle/sphere bounded by inscribing square/cube. The percentage of reachable voxel space (empirically evaluated) for all robots are given in Table I (Space Usage). The underlying probability distribution function, *i.e.*, the best $\widetilde{\mathcal{SV}}_G$, for the always on/off voxels are tightly peaked around the parameter values that would output 1/0. Regression models are able to learn such a function well and thus, our network easily learns always on/off voxels, yielding high accuracy predictions.

$\widetilde{\mathcal{SV}}_G$ also predicts swept volume geometry accurately. This is seen in Fig. 5 where $\mathrm{F}_1$ score against $\tau$ is plotted for all robots. Since $\mathrm{F}_1$ score is a measure of on-voxel prediction accuracy (regardless of off-voxel prediction accuracy), it is more directly correlated with the accuracy of the predicted geometry. Trained $\widetilde{\mathcal{SV}}_G$ for the base robot set show high quality with the highest score of 97.3% for PR at $\tau = 0.43$ and lowest score of 56.2% for Mico at $\tau = 0.9$. Precision can be increased at the cost of recall by increasing the threshold value, while recall can be increased at the cost of precision by lowering the threshold.

The $\mathrm{F}_1$ score of the different robots (Fig. 5) positively correlates to the amount of the space explored within the bounding volume V in the training data set (Table I, Space Usage). The availability of more *on* samples within V produces more learning signal and enhances the training result of $\widetilde{\mathcal{SV}}_G$. More exploration likely reduces the probability of false-negative predictions.

We also investigate how $\widetilde{\mathcal{SV}}_G$ quality is impacted by the probability threshold for on/off voxel classification, $\tau$. Fig. 5 shows that $\tau$ impacts the $\mathrm{F}_1$ score in a non-monotonic manner demonstrating peak values for $\tau \in [0.41, 0.48]$. (Although less pronounced, a similar trend is found for accuracy, Fig. 4.) This is expected as $\tau$ impacts the amount of false-positive/negative predictions as demonstrated in Fig. 6, where the relative voxel count of false-positive(blue)/negative(red) predictions to truth label is plotted as a function of $\tau$ for PR (top) and Mico (bottom). A lower threshold value corresponds to classifying a voxel to be on at lower occupation probability as output by the network, and hence will result in more false-positive prediction and vice versa. This trade-off between the amount of false-positive and negative predictions with $\tau$-values is seen in the inverse correlation between the false-positive/negative curves. More true-positive predictions are likely missed at higher $\tau$ values. This is reflected in the asymmetric trend of $\mathrm{F}_1$ score with respect to $\tau$ (Fig. 5).

Additionally, the false-positive/negative prediction curves show an insightful correspondence with the error computed for swept volume scalar value, $|\mathcal{SV}|$ (yellow dotted line in Fig. 6). This error value is the difference between the voxel count of predicted volume and the truth label. The value of $\tau$ for zero $|\mathcal{SV}|$-value and where false-negative/positive curves cross coincide; where false-negative and positive are balanced. The threshold value at this point, let it be $\tau_c$ can be useful in motion planning applications. For example, when using $\widetilde{\mathcal{SV}}_G$ in tasks where it is more important to avoid collisions, it is useful to favor less false-negative predictions. Thus, for the robots corresponding to Fig. 6 the threshold value should be $\tau < \tau_c = 0.41$ (Mico) and $\tau < \tau_c = 0.43$ (PR).

Training data availability (quality) also impacts learning quality of $\widetilde{\mathcal{SV}}_G$. This can be seen in the relative scale of prediction errors for PR (Fig. 6, top) and Mico (bottom). Out of the bounding volume of the training data, V, the reachable space of PR is 68.5% (highest) and 35.1% (lowest) for Mico (Table I). (CL is excluded due to a dissimilar sampling method, see Sec IV-B.) Therefore, PR's $\widetilde{\mathcal{SV}}_G$ received more learning signal from the training data compared to Mico, and thus PR's largest error value is much lower at 21.1%

| Robot | Training ($\times 10^3$ s) | $\widetilde{\mathcal{SV}}_G$ (ms) | $\mathcal{SV}_G$ (s) | $\Delta t$ |
|---|---|---|---|---|
| $\text{Kuka}_{10}$ | $0.5 \pm 0.02$ | $1.2 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| $\text{Kuka}_{15}$ | $0.7 \pm 0.04$ | $1.7 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| $\text{Kuka}_{20}$ | $1.4 \pm 0.07$ | $1.5 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| $\text{Kuka}_{25}$ | $1.9 \pm 0.08$ | $2.5 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| $\text{Kuka}_{30}$ | $3.3 \pm 0.20$ | $2.0 \pm 0.5$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| $\text{Kuka}_{80}$ | 725.3* | $3.9 \pm 0.5$ | $2.9 \pm 0.20$ | $5e^{-3}$ |
| Baxter | $1.5 \pm 0.04$ | $2.2 \pm 0.5$ | $1.0 \pm 0.04$ | $1e^{-2}$ |
| Sawyer | $2.1 \pm 0.05$ | $1.8 \pm 0.4$ | $1.0 \pm 0.04$ | $1e^{-2}$ |
| UR3 | $0.7 \pm 0.04$ | $1.7 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| UR5 | $1.5 \pm 0.06$ | $1.5 \pm 0.4$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| UR10 | $3.9 \pm 0.16$ | $2.1 \pm 0.5$ | $2.0 \pm 0.04$ | $5e^{-3}$ |
| Mico | $0.7 \pm 0.04$ | $1.7 \pm 0.4$ | $2.0 \pm 0.05$ | $5e^{-3}$ |
| Planar | $1.1 \pm 0.05$ | $1.6 \pm 0.3$ | $2.6 \pm 0.05$ | $4e^{-3}$ |
| PR | $3.1 \pm 0.20$ | $1.7 \pm 0.5$ | $2.1 \pm 0.03$ | $5e^{-3}$ |
| CL | $1.3 \pm 0.14$ | $2.0 \pm 0.5$ | $1.1 \pm 0.02$ | $1e^{-2}$ |

TABLE II

TIMES (TRAINING/PREDICTION) AND PARAMETER FOR TRAINING DATA GENERATION: (TRAINING) TIME TO LEARNING CONVERGENCE, ($\widetilde{\mathcal{SV}}_G$) SWEPT VOLUME PREDICTION TIME OF THE NETWORK POST-TRAINING, ($\mathcal{SV}_G$) AVERAGE TIME TO GENERATE A SWEPT VOLUME PREDICTION USING FIXED STEP SIZE ($\Delta t$). (NOTE THAT LEARNING (TRAINING/PREDICTION) AND SWEPT VOLUME TRAINING DATA GENERATION WERE PERFORMED ON DIFFERENT SYSTEMS DUE TO SYSTEM REQUIREMENTS.) *ONLY ONE RUN PERFORMED FOR $\text{KUKA}_{80}$.

compared to Mico's at 58.2% (Fig. 6).

### B. Learning and Prediction Time Evaluation

Here, we consider the computation time required to learn and predict $\widetilde{\mathcal{SV}}_G$ (Table II). First, consider the time to generate the training data for a single swept volume, $\mathcal{SV}_G$, and its parameter, the interpolation step size $\Delta t$ (columns 4 and 5). The time taken to generate a single swept volume is in the order of seconds for all the robots tested. We selected $\Delta t$ to be sufficiently small so that swept volume voxels are not skipped. However, $\Delta t$ can not be too small as smaller $\Delta t$ requires more time to generate training samples. Next, we discuss the time taken to train the network (column 2). The number of robot DOFs does not seem to have much of an effect on training time, as the 16-DOF Planar takes less time to train than the 7-DOF $\text{Kuka}_{20}$. However, grid size does have a significant effect on training time as seen by the increasing training times of the Kuka robots with increased grid size. Finally, consider the time taken by a trained network to predict $\widetilde{\mathcal{SV}}_G$ (column 3). Prediction times for all but $\text{Kuka}_{80}$ are between 1.2 and 2.5 ms on average with only a slight dependence on grid size. The differences are small because network structure varies only in input and output sizes. $\text{Kuka}_{80}$ predictions take 3.9 ms on average. This could be accounted for by the facts that $\text{Kuka}_{80}$ has a large grid size and a different network structure.

A common comparison would be to consider the time to compute $\widetilde{\mathcal{SV}}_G$ from the network and $\mathcal{SV}_G$. While these times are three orders of magnitude different, it should be noted that they were run on different systems due to optimized system requirements (detailed in Section IV).

### C. Spatial Distribution of Errors

| | Err. Within 1 Voxel (%) | | Err. Within 3 Voxels (%) | | Max Err. Dist. in # Voxels | |
|---|---|---|---|---|---|---|
| Robot | FP | FN | FP | FN | FP | FN |
| $\text{Kuka}_{10}$ | 98 | 99 | 100 | 100 | 3 | 3 |
| $\text{Kuka}_{15}$ | 93 | 97 | 100 | 100 | 5 | 3 |
| $\text{Kuka}_{20}$ | 95 | 95 | 100 | 100 | 4 | 3 |
| $\text{Kuka}_{25}$ | 88 | 91 | 100 | 100 | 6 | 4 |
| $\text{Kuka}_{30}$ | 86 | 87 | 100 | 100 | 8 | 4 |
| $\text{Kuka}_{80}$ | 50 | 49 | 91 | 96 | 28 | 7 |
| Baxter | 95 | 97 | 100 | 100 | 16 | 4 |
| Sawyer | 88 | 90 | 100 | 100 | 6 | 3 |
| UR3 | 96 | 98 | 100 | 100 | 3 | 3 |
| UR5 | 98 | 96 | 100 | 100 | 4 | 4 |
| UR10 | 94 | 93 | 100 | 100 | 7 | 3 |
| Mico | 95 | 99 | 100 | 100 | 3 | 3 |
| Planar | 35 | 34 | 79 | 77 | 23 | 19 |
| PR | 97 | 93 | 100 | 100 | 18 | 10 |
| CL | 89 | 100 | 100 | 100 | 6 | 3 |

TABLE III

DISTANCE OF FALSE-NEGATIVE (FN) AND FALSE-POSITVE (FP) VOXELS FROM THE $\mathcal{SV}_G$ BOUNDARY (ERR. DIST.). THE FIRST TWO COLUMNS SHOW THE PERCENTAGE OF FN AND FP ERRORS AT 1 VOXEL, AND MIDDLE TWO COLUMNS SHOW THE PERCENTAGE WITHIN 3 VOXELS FROM THE $\mathcal{SV}_G$ BOUNDARY, AND THE LAST TWO COLUMNS SHOW THE MAXIMUM ERROR DISTANCE IN THE NUMBER OF VOXELS (MAX ERR. DIST.)

For $\widetilde{\mathcal{SV}}_G$ to be a useful predictor, it is critical to understand where prediction errors occur so that we can minimize the impact during planning. Here, we analyze the spatial distribution of prediction errors by computing the shortest distance between falsely predicted voxels and the proximal surface voxel of the true volume, *the error distance*, via the Euclidean Distance Transform [31]. This metric is analyzed for both false-positive and false-negative voxels.

The majority of false predictions are found in the proximity of the true surface as seen in the error distance data for all robots in Table III. Thirteen of the fifteen robots have almost all errors occurring at a distance within three voxels and at least 86% within one voxel of the true surface (columns 1–4). However, $\widetilde{\mathcal{SV}}_G$ for $\text{Kuka}_{80}$ and Planar produce a relatively broader error distance distribution. The much higher training data resolution of $\text{Kuka}_{80}$ (0.025 m vs 0.1 m) means there are more voxels on and near the true surface, resulting in a broader-distributed error distance. Planar has a much higher number of DOFs (*i.e.*, 16 compared to 6–9 for all others), and the 3rd lowest space usage (Table I). These could contribute to the broader error distances (*e.g.*, larger input feature DOF could require a larger network and smaller space usage could lead to an insufficient learning signal).

Additional detail about rare false predictions that occur further from the true surface is found in columns 5 and 6, where the maximum distance of any error reported over test data is provided. For our base robot set, the max distance of error seems to be correlated with larger volume dimensions for the robots. For example, Planar has a volume V$= 1 \times 100 \times 100$ with max error distance of 23 (Table I,

V), both the largest for all the robots tested. Even though some errors occur far from the surface (Baxter, Planar, PR) these are rare and constitute a minute portion of the total error. For example, the maximum erroneous volume found for any robot is $6.57e^{-2}\%$ (UR3) of the total error. For the resolution study set, $\text{Kuka}_{10}$–$\text{Kuka}_{80}$, a similar trend for rare false predictions are seen. We note that while the max distance of 28 for $\text{Kuka}_{80}$ appear larger than the rest by a significant amount, this value is equivalent to only 7 voxels for the other robots since the width of the $\text{Kuka}_{80}$ voxel is 0.025 m instead of 0.1 m (all other robots).

### D. Impact of Data Resolution

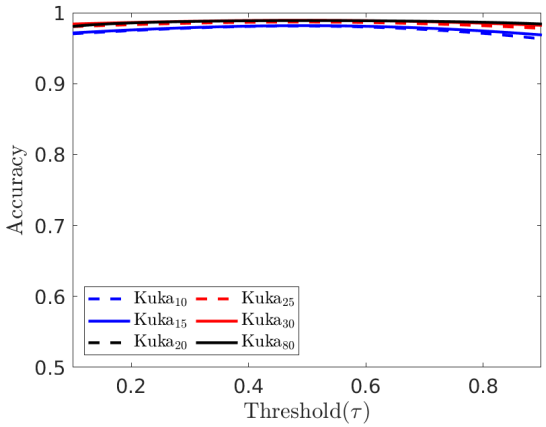


Fig. 7. Prediction accuracy as a function of output probability threshold ($\tau$) used to classify on/off voxels for the resolution study set.

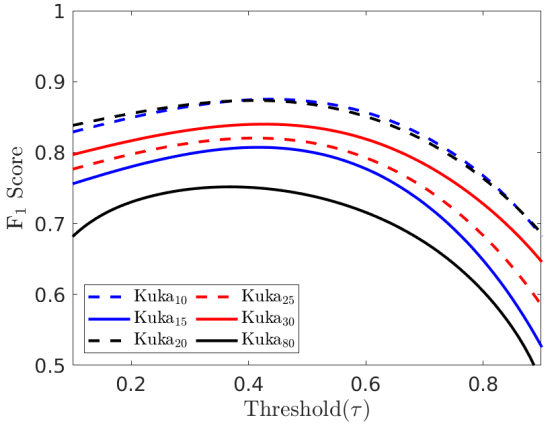


Fig. 8. $F_1$ score as a function of output probability threshold ($\tau$) used to classify on/off voxels for the resolution study set.

Finally, we evaluate the impact of training data resolution on $\widetilde{\mathcal{SV}}_G$ quality through accuracy and $F_1$ score for the resolution study set: $\text{Kuka}_{10}$, $\text{Kuka}_{15}$, $\text{Kuka}_{20}$, $\text{Kuka}_{25}$, $\text{Kuka}_{30}$, and $\text{Kuka}_{80}$. While we have explored these variants in previous sections, we now use the prior insights to gain understanding about the the differences in accuracy and $F_1$ score for these robots. Figure 7 shows that the higher the resolution, the better the accuracy, although with a slight change in the relative order at some values of $\tau$. Moreover, accuracy positively correlates with space usage (Table I), suggesting that the higher accuracy is likely due to more always-off voxels being present for higher resolution data as was seen with the base robot set.

The $F_1$ score better elucidates the differences in the resolution set, as shown in Fig. 8. Here we see that the highest resolution yields the lowest $F_1$ score and the lowest resolution yields the highest $F_1$ score. As explained in Section V-C, most of the error happens very close to the $\mathcal{SV}_G$ surface, and this has a higher impact at higher resolutions. The lower space usage (Table I) for the highest resolution, $\text{Kuka}_{80}$, also likely contributes to the lower $F_1$ score. A relatively smaller amount of learning signal from the lower space usage results in more erroneous predictions of true volume. Unlike the base robot set, $F_1$ score is not well correlated with space usage. While the robots in the resolution set with the higest and lowest space usage do show this trend, $\text{Kuka}_{10}$ and $\text{Kuka}_{80}$, there is no correlation for those in the middle. However, the space usage of this middle set is within about 10%, indicating little distinction. In contrast, $\text{Kuka}_{10}$ and $\text{Kuka}_{80}$ have almost a 30% space usage difference. It should be noted that for the five largest resolutions, $\text{Kuka}_{10}$ to $\text{Kuka}_{30}$, 100% of errors lie within three voxels of the surface (Table III). This indicates that there are few differences between these robots.

## VI. Conclusions

In this work, we evaluated the learned swept volume geometry function approximator, $\widetilde{\mathcal{SV}}_G$, for 10 robots with a variety of joint properties and DOF. $\widetilde{\mathcal{SV}}_G$ across all robots demonstrated the ability to accurately approximate the swept volume geometry given the start and end robot configurations as well as the occupation probability of each voxel. Characterizing the prediction errors revealed that errors are tightly distributed around the true volume surface. The deep geometry predictor was also shown to produce approximated swept volume geometries much faster than the Octree method; our $\widetilde{\mathcal{SV}}_G$ can realistically be integrated into online motion and task planning. Evaluating $\widetilde{\mathcal{SV}}_G$ quality against the voxel occupation probability threshold for on/off classification showed that a critical threshold value can be found. This critical value can be useful in tuning the network to favor false-negative/positive predictions depending on a motion planning application at hand.

Swept volume geometry predictions are expected to prove very useful in motion and task planning applications. For these applications, further studies in the nature of the erroneous predictions in relation to the robot's body as well as the ability of the network to predict the geometry in restricted configuration spaces, e.g., lower dimensional sub-manifolds, may prove to be critical.

## VII. Acknowledgement

We would like to acknowledge Hao-Tien Lewis Chiang, Waymo Research, and Aleksandra Faust, Google Brain, for their helpful ideas and discussion on this work.

## References

[1] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Auto. Robot Vehicles*, 1990, pp. 259–271.

[2] F. Schwarzer, M. Saha, and J.-C. Latombe, "Exact collision checking of robot paths," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2004, pp. 25–41.

[3] J. J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004, pp. 3993–3998.

[4] P. G. Xavier, "Fast swept-volume distance for robust collision detection," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2. IEEE, 1997, pp. 1162–1169.

[5] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. Konidaris, "Robot motion planning on a chip," in *Proc. Robotics: Sci. Sys. (RSS)*, 2016.

[6] H. Täubig, B. Bäuml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1585–1592.

[7] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 427–439, 2012.

[8] H.-T. L. Chiang, A. Faust, S. Sugaya, and L. Tapia, "Fast swept volume estimation with deep learning," in *Algorithmic Foundations of Robotics XIII*, M. Morales, L. Tapia, G. Sanchez-Ante, and S. Hutchinson, Eds. Springer, 2018, pp. 52–68.

[9] L. Gueta, R. Chiba, T. Arai, T. Ueyama, J. Rubrico, and J. Ota, "Compact design of a redundant manipulator system and application to multiple-goal tasks with temporal constraint," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 11, pp. 1–13, 4 2017.

[10] C. Ekenna, D. Uwacu, S. Thomas, and N. M. Amato, "Improved roadmap connection via local learning for sampling based planners," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2015, pp. 3227–3234.

[11] K. Abdel-Malek, J. Yang, D. Blackmore, and K. Joy, "Swept volumes: foundation, perspectives, and applications," *International Journal of Shape Modeling*, vol. 12, no. 01, pp. 87–127, 2006.

[12] A. Gaschler, R. Petrick, T. Kröger, O. Khatib, and A. Knoll, "Robot task and motion planning with sets of convex polyhedra," in *Proc. Robotics: Sci. Sys. (RSS)*, 2013.

[13] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical planning in the now," in *Proc. Int. Conf. Artif. Intel.*, 2010, pp. 33–42.

[14] R. R. Martin and P. C. Stephenson, "Sweeping of three-dimensional objects," *CAD*, vol. 22, no. 4, pp. 223–234, 1990.

[15] M. Campen and L. Kobbelt, "Polygonal boundary evaluation of Minkowski sums and swept volumes," in *Computer Graphics Forum*, vol. 29, no. 5, 2010, pp. 1613–1622.

[16] S. Abrams and P. K. Allen, "Swept volumes and their use in viewpoint computation in robot work-cells," in *Proc. of IEEE Int. Symp. on Assembly and Task Planning*, 1995, pp. 188–193.

[17] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha, "Fast swept volume approximation of complex polyhedral models," *Computer-Aided Design*, vol. 36, no. 11, pp. 1013–1027, 2004.

[18] J. C. Himmelstein, E. Ferre, and J.-P. Laumond, "Swept volume approximation of polygon soups," *IEEE Trans. on Autom. Sci. and Eng.*, vol. 7, no. 1, pp. 177–183, 2010.

[19] A. Von Dziegielewski, M. Hemmer, and E. Schömer, "High precision conservative surface mesh generation for swept volumes," *IEEE Trans. on Autom. Sci. and Eng.*, vol. 12, no. 1, pp. 183–191, 2015.

[20] S. Abrams and P. K. Allen, "Computing swept volumes," *The Journal of Visualization and Computer Animation*, vol. 11, no. 2, pp. 69–82, 2000.

[21] H.-T. L. Chiang, J. E. G. Baxter, S. Sugaya, M. R. Yousefi, A. Faust, and L. Tapia, "Fast deep swept volume estimator," *IJRR*, p. to appear, 2020.

[22] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3D mesh approximate convex decomposition," in *Int. Conf. on Image Processing (ICIP)*. IEEE, 2009, pp. 3501–3504.

[23] M. Gadelha, S. Maji, and R. Wang, "3D shape induction from 2D views of multiple objects," *2017 International Conference on 3D Vision (3DV)*, pp. 402–411, 2016.

[24] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2463–2471.

[25] A. Dai, C. Ruizhongtai Qi, and M. NieBner, "Shape completion using 3D-encoder-predictor cnns and shape synthesis," 2017, pp. 6545–6554.

[26] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2013, pp. 1321–1326.

[27] A. K. V. Badrinarayanan and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[28] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520–1528.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–15, 2014.

[30] J. Nam, J. Kim, E. L. Mencia, I. Gurevych, and J. Furnkranz, "Large-scale multi-label text classification — revisiting neural networks," in *Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 437–452.

[31] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, *A General Algorithm for Computing Distance Transforms in Linear Time*. Boston, MA: Springer US, 2000, pp. 331–340.