Accepted for publication in IEEE International Conference on Robotics and Automation (ICRA), June 2021

Learning Sampling Distributions Using Local 3D Workspace Decompositions for Motion Planning in High Dimensions

Constantinos Chamzas, Zachary Kingston, Carlos Quintero-Peña,
Anshumali Shrivastava, and Lydia E. Kavraki

***Abstract*— Earlier work has shown that reusing experience from prior motion planning problems can improve the efficiency of similar, future motion planning queries. However, for robots with many degrees-of-freedom, these methods exhibit poor generalization across different environments and often require large datasets that are impractical to gather. We present SPARK and FLAME, two experience-based frameworks for sampling-based planning applicable to complex manipulators in 3D environments. Both combine samplers associated with features from a workspace decomposition into a global biased sampling distribution. SPARK decomposes the environment based on exact geometry while FLAME is more general, and uses an octree-based decomposition obtained from sensor data. We demonstrate the effectiveness of SPARK and FLAME on a real and simulated Fetch robot tasked with challenging pick-and-place manipulation problems. Our approaches can be trained incrementally and significantly improve performance with only a handful of examples, generalizing better over diverse tasks and environments as compared to prior approaches.**

I. Introduction

For many high-dimensional systems, sampling-based methods have proven very successful at efficient motion planning [1]. However, as noted by [2], they are sensitive to "challenging regions", such as narrow passages in the search space, which have a low probability of being sampled. The existence of such "challenging regions" hinders the performance of sampling-based planners.

Many methods have improved the performance of sampling-based planners by using *experience* from prior motion planning problems, e.g., [3], [4]. In particular, experience can be used to learn how to explore the "challenging regions" of the search space. However, for high dimensional robots, learning-based methods typically face difficulties in generalizing even between similar environments [5]–[7].

This work presents two experience-based planning frameworks—one using exact geometry (SPARK) and one using sensor data (FLAME)—for high DOF robots in 3D environments. Both frameworks build on the decomposition paradigm introduced by [5]. That work introduced the idea of *local primitives*, which capture important local workspace features. Each primitive is associated with a *local sampler*, which produces samples in "challenging regions" created by the primitive, and are stored in an experience database. The database is updated offline by adding experiences from past planning problems in an incremental manner. Given a planning query, similar local primitives to those in the current workspace are retrieved; their local samplers are combined and used by a sampling-based planner. In such a framework several questions arise on the definition and use of local primitives. [5] required *a priori* knowledge of local primitives and was designed for 2D environments with circular obstacles.

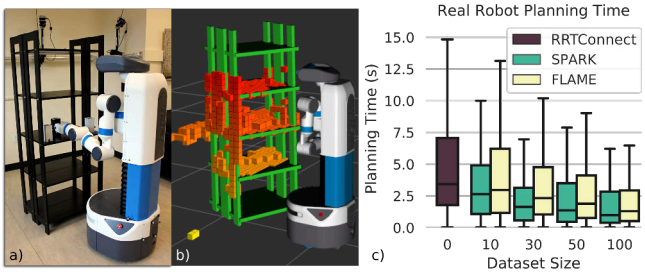

Fig. 1. The proposed sampling-biasing frameworks applied in a real challenging motion planning problem. **a)** The 8 DOF Fetch robot must reach into a shelf to grasp one of the randomly positioned blocks. Between problems, the base pose varies by up $\pm 90^\circ$ in angular position and up to ± 10 cm along the X- and Y-coordinates. **b)** A VICON camera system was used to gather object poses for SPARK. The Fetch's depth camera was used to build an octomap [8] for FLAME which can retrieve relevant experience even from partial noisy octomaps. **c)** Both improve RRT-Connect even with few given examples.

The contributions of this paper are the following. We present SPARK which scales to 3D environments when exact geometric information is available and FLAME which relies only on sensor information. In order to develop these frameworks, we revisit and enrich the framework of [5] by introducing novel local primitives, distance functions, and utilizing efficient retrieval structures. Furthermore, we propose a new *criticality test* that enables incremental training of SPARK and FLAME from novel scenes. Through the synergistic combination of the introduced components, we achieve real-time retrieval and training of both frameworks. Finally we validate our methods on a suite of simulated and real robot experiments with the Fetch robot, an 8 degree-of-freedom (DOF) manipulator. Compared to prior work, our methods exhibit significantly better performance over diverse environments and tasks with relatively few examples. The datasets and the implementation of SPARK and FLAME are open-sourced [1].

II. Problem Statement

Consider a robot in a workspace $\mathcal{W}$. A robot configuration x is a point in the configuration space ($\mathcal{C}$-space), $x \in \mathcal{C}$.

CC is supported by NSF-GRFP 1842494 and ZK is supported by NSTRF 80NSSC17K0163. This work is also was supported in part by NSF 1718478, and Rice University Funds.

CC, ZK, CQP, AS and LEK are with the Department of Computer Science, Rice University, Houston, TX, USA {chamzas,zak,carlosq,anshumali,kavraki}@rice.edu

[1] https://github.com/KavrakiLab/pyre

Obstacles in workspace induce $\mathcal{C}$-space obstacles where the robot is in collision. We are interested in finding a collision-free path (p), from start to goal as efficiently as possible.

Sampling-based planners are widely used for motion planning [1], [9]. These planners randomly sample to explore the $\mathcal{C}$-space. However, the performance of sampling-based planners is degraded when "challenging regions" exist in the $\mathcal{C}$-space [2]. These regions, introduced by $\mathcal{W}$ and the robot geometry, create difficult-to-explore areas in the $\mathcal{C}$-space due to their "low visibility" relative to the entire $\mathcal{C}$-space. It is understood [10], [11] that sampling in such regions can significantly improve performance. Thus, the problem we consider is to generate a biased sampler $P(x|\mathcal{W})$ which produces samples in "challenging regions.". In other words, we want to learn a mapping from $\mathcal{W}$ to samples in "challenging regions."

III. Related Work

One way to bias sampling is with rejection sampling, which draws uniform samples but only accepts them if a geometric test is passed, e.g., bridge sampling [2] or Gaussian sampling [12]. However, rejection sampling is inefficient, possibly drawing many samples before a valid one is found. Some approaches attempt to directly sample the "challenging regions," such as medial axis sampling [13] or free-space dilation [14], but these distributions become difficult to compute as the state space increases in dimension.

Rather than using fixed distributions, many methods [15], [16] *adapt* sampling online based on collision information, such as utility-guided sampling [17], toggle-PRM [18] or hybrid-PRM [19]. However, these methods do not transfer knowledge between planning queries. Our methods bias based on prior motion plans, and thus is complementary to adaptive sampling methods.

Another category of techniques improves planning by learning problem invariants. Examples include using sparse roadmaps [3], informed lazy evaluations [20], path databases [21]–[23], or biased distributions [24]. However, these methods do not use workspace information. Thus, changes in the environment force these methods to repair now invalidated information, reducing performance.

On the other hand, many methods do use workspace features to guide sampling—typically, exact [25], [26] or inexact [27] workspace decompositions are used. For example, the authors of [28] use balls to decompose the workspace. However, to generate samples in "challenging regions," these approaches require a mapping from workspace to $\mathcal{C}$-space, reliant on the robot's inverse kinematics. Thus, these methods are typically limited to free-flying or low-dimensional robots, as manipulator kinematics can become complex. The proposed methods also use workspace decompositions (both an exact and inexact one), but instead *learn* the inverse mapping from workspace to $\mathcal{C}$-space.

Many recent techniques use deep learning to learn the mapping from workspace to $\mathcal{C}$-space, such as using neural networks to bias sampling [4], [6], [29], [30] or to guide planners [31]–[33]). However, due to the inherent discontinuity of motion planning [34], [35], this mapping is generally discontinuous, limiting the applicability of these methods [5], [7], [36]. That is, small changes in the workspace result in substantial (possibly discontinuous) changes in $\mathcal{C}$-space. Thus, these approaches usually apply only to manipulators with low workspace variance [4], [29]–[31].

On the other hand, a category of methods that addresses the discontinuity problem, are mixture model methods and similarity-based methods. Mixture model methods [7], [37] learn different functions for different discontinuous regions but are limited by the fixed number of mixtures. Similarity-based methods [5], [38], [39], which are most similar to ours, are usually non-parametric techniques that retrieve relevant information (e.g., "challenging regions" representations) based on workspace similarity. Retrieval is by design discontinuity-sensitive since multiple different "challenging regions" can be retrieved for similar workspaces. However, [38] is limited to free flying robots [39] addresses trajectory optimization-based problems rather than sampling and [5] is limited to 2D planar environments with circular objects. Our work shares the retrieval idea with these methods but applies to general 3D workspaces with realistic robots.

IV. Methodology

We introduce two experience-based planning frameworks for arbitrary 3D environments and manipulators, SPARK and FLAME. SPARK requires an exact representation of the environment's geometry, while FLAME only uses sensor information represented as an octomap [8].

Consider the Fetch robot reaching into a deep shelf, as shown in Fig. 1. The arrangement of the workspace obstacles constricts the manipulator's movement, making the motion planning problem challenging. Both frameworks decompose the workspace into *local primitives* to capture local workspace features that create "challenging regions" in $\mathcal{C}$-space, e.g., the shelf's sides that create a narrow region for reaching. Our frameworks learn by using a *criticality test*, which associates "critical" configurations from every successful motion plan to relevant local primitives in the current workspace. For example, the criticality test should capture that configurations of the Fetch's arm deep in the shelf, should be associated with the sides of the shelf. These configurations are used to create a *local sampler*—a biased sampling distribution focused on "challenging regions" created by the associated local primitive. During learning, these primitive-sampler pairs are stored in a spatial database (Alg. 1). When inferring for a new problem, relevant local primitive-sampler pairs are retrieved from the database; local samplers are then combined into a global biased sampler (Alg. 2).

A. Overall Framework

Both SPARK and FLAME define their own decomposition of the workspace into local primitives, a criticality test for these primitives, and a metric to compare local primitives. The database that stores local primitives, their associated local samplers, and the synthesis of the global sampler is the same in SPARK and FLAME.

Algorithm 1: Learning

Input : Workspace $\mathcal{W}$, Path p, Database $\mathcal{B}$
Output : Database $\mathcal{B}$

1 Decompose $\mathcal{W}$ to $\mathcal{L} \leftarrow \{\ell_1, \ldots, \ell_N\}$
2 **foreach** $\ell_j \in \mathcal{L}$ **do**
3 CRITICAL $\leftarrow \emptyset$
4 **foreach** $x \in p$ **do**
5 **if** ISCRITICAL$(x,\ \ell_i)$ **then**
6 CRITICAL $\leftarrow$ CRITICAL $\cup\ x$
7 Generate $P_j\left(x \mid \ell_j\right)$ from CRITICAL
8 Add $\langle \ell_j, P_j\left(x \mid \ell_j\right)\rangle$ in $\mathcal{B}$
9 **return** $\mathcal{B}$

1) Local Primitives: Local primitives are workspace features that come from any valid decomposition of the workspace. Each local primitive attempts to capture the workspace features that create "challenging regions" in the $\mathcal{C}$-space. The definition of local primitives is a key component of the algorithm. The local primitives for SPARK and FLAME are given in their respective sections, and are shown in Fig. 2 and Fig. 3. We denote local primitives with ℓ. From a given workspace $\mathcal{W}$, N local primitives are created from a workspace decomposition such that $\bigcup_{i=1}^{N} \ell_i \subseteq \mathcal{W}$ (line 1 in Alg. 1, line 1 in Alg. 2).

2) Local Samplers: Each local primitive ℓ_i is associated with a local sampler, which should produce samples in the "challenging regions" created by the associated local primitive. A local sampler $P_i\left(x \mid \ell_i\right)$ is a generative distribution over a set of M_i "critical" configurations $X_i = \{x_{i1}, \ldots, x_{iM_i}\}$ gathered from past experience. Due to the potential complexity of this distribution and its natural multi-modality, we represent the local sampler as a Gaussian mixture model (GMM), similar to [24]. Contrary to [24], we do not use expectation maximization to calculate the parameters of the GMM. Instead, we place one mixture for each configuration x_{ij} in the local sampler P_i with a fixed covariance σ (line 7 in Alg. 1):

$$P_i\left(x \mid \ell_i\right) = \frac{1}{M_i} \sum_{j=1}^{M_i} \mathcal{N}(x_{ij}, \sigma).$$

We choose uniforms weights—all mixtures are equiprobable.

3) Criticality Test: In 3D local primitives may or may not create "challenging regions," and finding if a configuration belongs to such a region is not straightforward. The authors of [40] chose regions with maximum path density while [41] and [6] identified "challenging regions" using graph properties of a roadmap. However in our setting only finding these regions is not enough, since they must also correspond to a local primitive. To overcome this problem, and depart from earlier work, we introduce a *criticality test* (ISCRITICAL, line 4 in Alg. 1), which associates key configurations from solution paths to relevant local primitives. The intuition is that a configuration in a critical region will be close to a $\mathcal{C}$-space obstacle, which often means that some part of the robot will be close to a workspace obstacle. Both SPARK and FLAME define their own criticality test, which enables learning incrementally without *a priori* knowledge.

Algorithm 2: Inference

Input : Query Workspace $\mathcal{W}$, Database $\mathcal{B}$
Output : Sampler $P\left(x \mid \mathcal{W}\right)$

1 Decompose $\mathcal{W}$ to $\mathcal{L} \leftarrow \{\ell_1, \ldots, \ell_N\}$
2 **foreach** $\ell_j \in \mathcal{L}$ **do**
3 Retrieve all $\{P_i\left(x \mid \ell_i\right) \mid d(\ell_i, \ell_j) \leq d_{\text{radius}}\}$
4 Synthesize $P\left(x \mid \mathcal{W}\right)$ from all $P_i\left(x \mid \ell_i\right)$
5 **return** $P\left(x \mid \mathcal{W}\right)$

4) Distance Function: Given a new motion planning problem, the workspace is decomposed into a set of local primitives. Based on a *distance* metric over local primitives $d(\ell_i, \ell_j)$, we search for similar local primitives in the existing database $d(\ell_i, \ell_j) \leq d_{\text{radius}}$ and retrieve their associated local samplers (line 3 in Alg. 2). Both SPARK and FLAME define a distance metric over their local primitives.

Quick retrieval of relevant local primitives is essential to our method. We use GNAT [42], a data structure with logarithmic scaling for nearest-neighbor retrieval in high-dimensional metric spaces. GNAT also enables fast rejection of queries that do not exist within the database, which is useful as many local primitives are irrelevant e.g., outside the reachable workspace.

5) Global Sampling: Given a set of K retrieved local samplers, they are combined in a composite GMM (line 4 in Alg. 2):

$$P\left(x \mid \mathcal{W}\right) \approx \frac{1}{K} \sum_{i=1}^{K} P\left(x \mid \ell\right) = \frac{1}{K} \sum_{i=1}^{K} \frac{1}{M_i} \sum_{j=1}^{M_i} \mathcal{N}(x_{ij}, \sigma).$$

To avoid redundancy as local samplers may be the same, a local sampler is used only if its unique. To retain probabilistic completeness, there is a chance $\lambda > 0$ to sample uniformly rather than from the GMM. If no local samplers are retrieved the biased sampler defaults to uniform sampling, i.e., $\lambda = 1$.

B. SPARK

1) Local Primitives: In SPARK, local primitives are defined as pairs of box obstacles ω^a and ω^b, which have the following parameters:

$$\ell_i = \{\omega_i^a, \omega_i^b\} = \{T_i^a, s_i^a, T_i^b, s_i^b\},$$

where T, s respectively denote the pose and size—a 3D vector for the scale of each dimension—of each box, for a total of 10 parameters per box. Non-box objects are approximated by bounding boxes. Boxes that are too far from each other (see Sec. IV-B.3) are not considered valid local primitives. An example of pairs of local primitives can be seen in Fig. 2.

Intuitively, pairs of objects create "challenging regions" in the configuration space, as they constrict space in the workspace. Note that one or many (e.g., 3) objects per local primitive are also viable options. However, single objects are less discriminative than two and thus recall extraneous information. In a similar vein, considering many objects combinatorially increases the number of primitives in a scene. As we show in Sec. V, using pairs of objects provides good results in our tested scenarios.

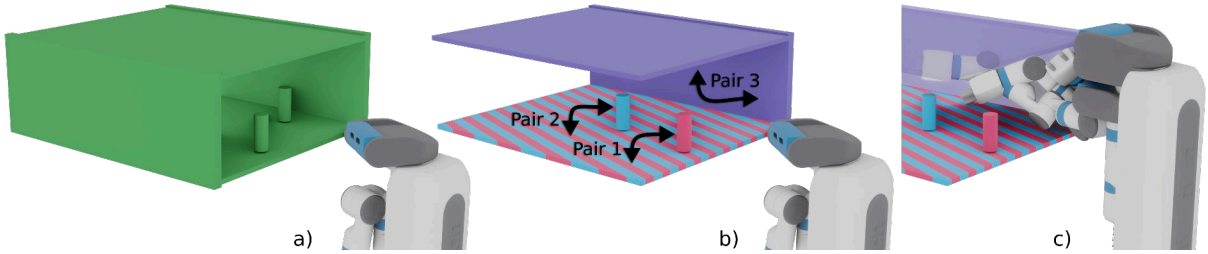

Fig. 2. An example of SPARK, our experience-based planning framework that decomposes the workspace into pairs of known obstacles. **a)** The global workspace, composed of a set of boxes and cylinders. **b)** Three local primitives (different pairs are shown with blue, magenta, pink color) from the SPARK decomposition. Note that the bottom of the shelf is shared between two primitives. **c)** A set of superimposed samples drawn from the local samplers.

2) Criticality Test: To check if a configuration is associated with a local primitive, we find the minimum distance between the robot at that configuration with the pair of obstacles. For every shape in the robot's geometry, the distance to both objects in the local primitive is measured with the Gilbert-Johnson-Keerthi (GJK) algorithm [43]. If the minimum distance is smaller than a threshold d_{clust}, that configuration is related to that local primitive.

3) Distance: To measure the distance between poses in $\text{SE}(3)$, we use [44]:

$$d_{\text{SE}(3)}\left(T_i, T_j\right) = w_T \left\|t_i - t_j\right\| + (1 - w_T)\left(1 - \langle q_i, q_j \rangle^2\right),$$

where t, q respectively denote translation and orientation (a quaternion) of a pose T, w_T is a chosen weight, $\|\cdot\|$ is the Euclidean norm and $\langle \cdot, \cdot \rangle$ is the inner product. To measure distance between boxes, we use a sum of metrics:

$$d_\omega(\omega_i, \omega_j) = w_s d_{\text{SE}(3)}\left(T_i, T_j\right) + (1 - w_s)\left\|s_i - s_j\right\|,$$

where w_s is a chosen weight. Local primitives for SPARK are all the pairs of boxes $d_\omega(\omega_i, \omega_j)$ with distance less than d_{pairs}. To compare local primitives, we use a metric that is the minimum distance between the boxes in each primitive:

$$d(\ell_i, \ell_j) = \min \begin{cases} d_\omega(\omega_i^a, \omega_j^a) + d_\omega(\omega_i^b, \omega_j^b) \\ d_\omega(\omega_i^a, \omega_j^b) + d_\omega(\omega_i^b, \omega_j^a) \end{cases}$$

C. FLAME

1) Local Primitives: In FLAME, local primitives are defined as "local" occupancy grids, or *octoboxes*, which are obtained from a global *octree*. An octree is a hierarchical volumetric tree that efficiently encodes obstacles in the workspace. At each level, the octree contains voxels—descending a level of the tree, these voxels are split in half along each of its axes, each giving 8 subvoxels. A voxel contains occupancy information at its resolution; a voxel is either occupied or free. Voxels at higher levels can compact information from lower levels—if all subvoxels are either occupied or free, then the tree does not need to descend further. FLAME is useful for application on real robots as octrees [8] are a common approach used to represent sensor data.

We define *octoboxes* as occupancy grids that correspond to intermediate nodes of an octree. Here, octoboxes contain the voxels of the two lowest levels of the octree, that is, 64 voxels as a $4 \times 4 \times 4$ 3D occupancy grid.

To decompose the workspace, an octree is built relative to the robot base frame that captures the occupancy of the workspace. Non-empty octoboxes are quickly generated by traversing the octree at a specific depth. We also consider the pose T of the center of the octobox relative to the robot base frame. An example is shown in Fig. 3. Local primitives in FLAME are $\ell_i = \{T_i, b_i\}$, where T is the pose of the octobox and b is a 64-bit occupancy grid vector, where each bit corresponds to an occupied (1) or free (0) voxel.

2) Criticality Test: A configuration x_i is associated with a local primitive if the robot's geometry at configuration x_i intersects the bounding box of the given octobox. Note that a configuration may be associated with many local primitives.

3) Distance: To compare two octoboxes, we simply check if the two octoboxes are the same. It may seem overly pessimistic to look for precise octobox matches, but our empirical results show that this simple metric yields good results. One possible explanation is, since the criticality test assigns a configuration to many different local primitives, only one primitive needs to be recovered to recall the relevant configuration. The distance metric is:

$$d(\ell_i, \ell_j) = \begin{cases} 0 & b_i = b_j, T_i = T_j \\ \infty & \text{otherwise} \end{cases}$$

V. EXPERIMENTS

We evaluate the performance of SPARK and FLAME in the following varied and realistic scenarios using a Fetch (8-DOF) robot manipulator. We believe both frameworks can also be used with mobile/flying robots, but focus on manipulators since we consider them more challenging for learning methods. The "Small Shelf" environment (Fig. 4a) with 3 cylindrical objects, the "Large Shelf" environment with 7-9 cylindrical objects (Fig. 6a), the "Real Shelf" (Fig. 1) with 3 box objects and the "Box Table" with mutliple random objects (Fig. 5a). In each environment, we vary both the pose of "small" objects (cylinders, boxes) and more importantly the pose of "big" objects(shelves, box, table) relative to the robot. The "big" objects variations of "Small shelf" are shown in Fig. 4a) and described in Fig. 1a), Fig. 5a), and Fig. 6a) for the other scenarios. See the attached video for more details.

We consider these variations of environments highly challenging for learning-based motion planning frameworks for two reasons. First, the underlying motion planning problem is hard since it exhibits many "challenging regions." Second,

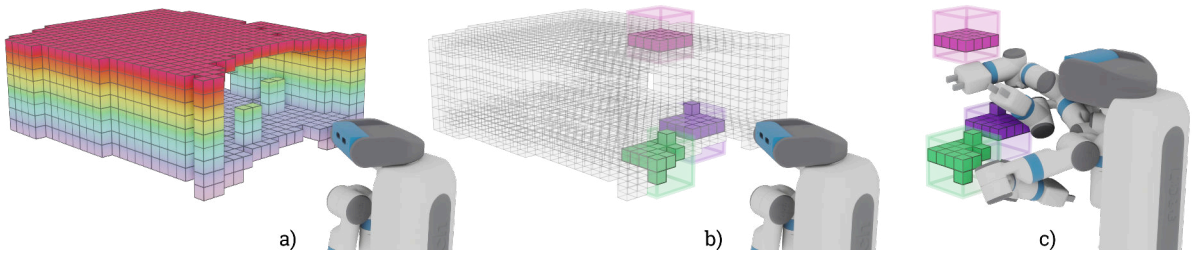

Fig. 3. An example of FLAME, our experience-based planning framework that uses a decomposition of an octomap [8] workspace gathered from sensing data. **a)** The global workspace as an octomap. **b)** Three local primitives (highlighted in color) from the FLAME decomposition. Local primitives are $4 \times 4 \times 4$ occupancy grids derived from the global octomap. **c)** A set of superimposed samples drawn from the local samplers.

TABLE I
SPARK AND FLAME HYPERPARAMETERS

Parameter	Symbol	Value
Sampling Variance	σ	0.2
Sampling Bias	λ	0.5
Retrieval Radius	d_{radius}	0.1
SPARK Clustering Distance	d_{clust}	0.15
SPARK Pair Distance	d_{pairs}	0.2
SPARK Pose Weight	w_T	0.75
SPARK Size Weight	w_s	0.5

even small changes in the relative position of the objects can cause discontinuous changes in the configuration space obstacles, making it hard to learn functions that map from workspace to $\mathcal{C}$-space [7], [35]. For example, a small change in the relative angle of the shelf to the robot might require a solution that goes on the left side of the robot, versus the right side. Such diversity is realistic, as mobile manipulators can approach objects from many different angles and positions.

We test two tasks: a "pick" and a "place" task. In the "pick" task, the Fetch plans between its stow position and the farthest object on the shelf. In the "place" task in the "Large Shelf" environment, the Fetch starts rigidly grasping the object farthest back on the top or bottom shelf and places the object in the back of the middle shelf. In the "place" task in the "Box Table" environment, the Fetch starts grasping the cube inside the box and place randomly on the table.

We implemented our framework using the Open Motion Planning Library (OMPL) [45] and *Robowflex* [46]. Note that SPARK and FLAME both generate sampling distributions and must be used by a sampling-based planner. For all experiments and frameworks, RRT-Connect [47] is used with default settings unless noted otherwise. All the experiments were run on an Intel® i7-6700 with four 4.00GHz cores and 32 GB of memory. The hyperparameters for SPARK (7) and FLAME (3) are given in Table I and were the same for all experiments. These are reasonable defaults for our methods, found heuristically. The methods performance was empirically observed to be robust over a wide range of values.

A. Generalization

Fig. 4b presents timing results for a "pick" task in "Small Shelf" over three types of variation, XY, XYZ, and $XYZ\Theta$ as shown in Fig. 4a. We compare against RRT-Connect, an experience-based approach that uses a path database (THUNDER) [3], and a conditional variational autoencoder framework (CVAE) [4]. THUNDER stores previous paths as a sparse roadmap—new problems are solved by searching the roadmap for a valid path; if no path is found (possibly due to changes in the environment), a planner is used to repair the roadmap or find a new path from scratch. The OMPL version of THUNDER was used within *Robowflex*.

CVAE learns a mapping from workspace to "interesting samples" in $\mathcal{C}$-space. To infer new samples the latent space of the CVAE can be sampled and decoded into $\mathcal{C}$-space, providing promising samples to a sampling-based planner (here, RRT-Connect is used). We use the provided implementation of CVAE [4]. We test two variants of CVAE: "CVAE w/o Task", which has the same input as SPARK albeit with a fixed number of obstacles, and "CVAE", which includes the start and goal as recommended in [4]. The latent space dimension was increased as proposed by [6] for our high DOF problems. The same training data (500 paths) were given to CVAE as SPARK which is approximately 5000 training samples.

All methods show marked improvement when the variance is low. However, only SPARK, FLAME, and CVAE retain performance as the axes of variation increases (Fig. 4b). As environment variance grows, THUNDER's sparse roadmap becomes less informative and falls back to planning from scratch. Similarly, the performance of "CVAE w/o Task" degrades as diversity increases (possibly due to discontinuities). CVAE greatly benefits from the addition of the start and goal, indicating it is highly informative for this task. Increasing the size of training examples did not improve the performance of CVAE Finally, note that SPARK and FLAME have lower variance in solution time, showing both become more effective at solving the "harder" problems in the test set.

B. Convergence and Scalability

Fig. 4c, Fig. 5b and Fig. 6b present convergence results for SPARK and FLAME for the "pick" task in the "Small Shelf" and the place task for "Large Shelf,""Box table" environments. SPARK and FLAME both achieve significant improvement in average performance given only a few examples and quickly converge ($\approx$ 300 for "Small Shelf," " Box Table" and $\approx$ 600 for "Large Shelf"). SPARK has better asymptotic performance (especially in "Box Table") than FLAME. This is expected, as FLAME uses the less informative but more general decomposition. Fig. 5c shows the database retrieval time versus the size of the dataset using naïve linear search and GNAT-based search. The number of local primitives for

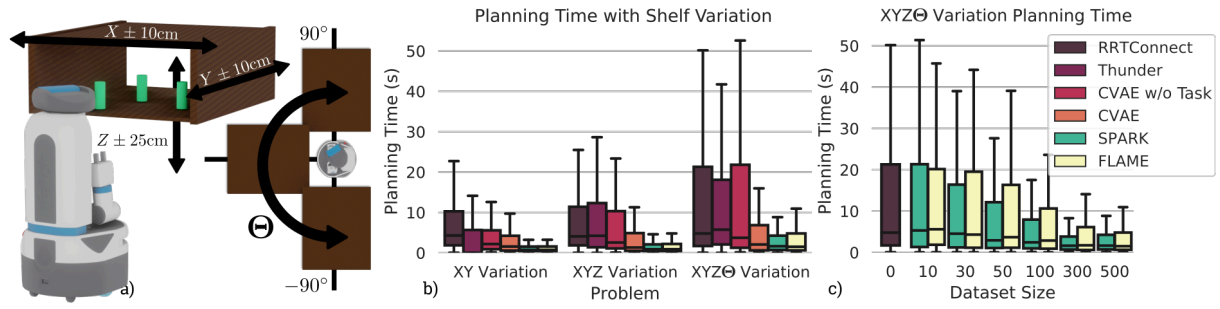

Fig. 4. **a)** The "Small Shelf" environment with a "pick" task. As shown, in this experiment, the orientation of the shelf relative to the robot varies up to $\pm 90°$, in its X- and Y-coordinates by ± 10cm, and by ± 25cm in it Z-coordinate. **b)** Average planning time (including retrieval) versus the axes of variances. 500 training and 100 test environments were sampled for each, with a timeout of 60 seconds. **c)** Average planning time (including retrieval) versus the size of the dataset, tested on 100 sampled environments with a timeout of 60 seconds.

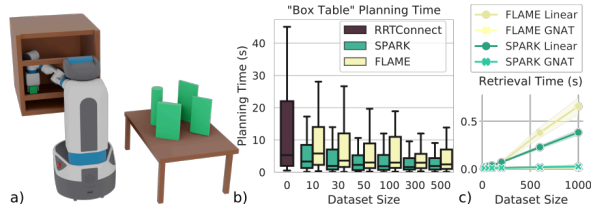

Fig. 5. **a)** The "Box Table" environment with a "place" task. The table and box vary relative to the robot in angular placement $\pm 30°$ and position of X- and Y-coordinates by ± 10cm. **b)** Plot of average planning time (including retrieval) versus the size of the dataset, tested on 100 sampled environments with a timeout of 120 seconds. **c)** Mean local primitive retrieval time versus the size of the dataset.

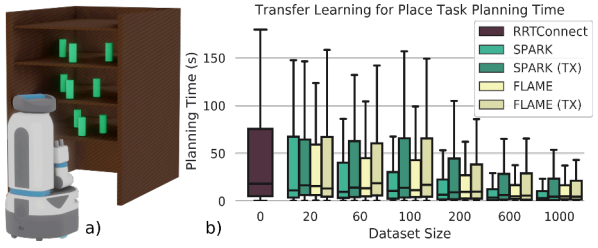

Fig. 6. **a)** The "Large Shelf" environment with a "pick" and "place" task. Three stacked shelves are generated similar to the "Small Shelf" environment (Fig. 4a). The stacked shelves vary relative to the robot in angular placement $\pm 90°$ and position of X- and Y-coordinates by ± 10cm. **b)** Plot of average planning time (including retrieval) versus the size of the dataset, tested on 200 sampled environments with a timeout of 300 seconds.

1000 training examples is $\approx 12,000$ for SPARK and $\approx 17,000$ for FLAME. GNAT scales much better than naive linear search. Although the retrieval overhead is negligible, we expect that it would affect performance if the database grows too large.

C. Transfer

Fig. 6 presents transfer learning results for SPARK and FLAME on a "place" task in the "Large Shelf" environment. Here, we show how SPARK and FLAME can learn from examples in one task and apply their experience to a different, harder task. In Fig. 6, a baseline of SPARK and FLAME trained and tested both on "place" tasks (labeled as SPARK and FLAME) is compared to SPARK and FLAME trained on "pick" tasks and tested on the "place" task (labeled as SPARK (TX) and FLAME (TX)). Note that the "place" task changes the robot's geometry by attaching an object in the end-effector's grasp, making the planning problem harder. Even when using experience gathered on a different, easier motion planning problem both SPARK and FLAME demonstrate significant performance gains, successfully transferring their experience. We hypothesize that FLAME shows better performance in transfer learning as it uses an approximate decomposition, and thus can find more relevant experience than SPARK.

D. Real Robot

Fig. 1 shows timing results for SPARK and FLAME applied to a real robot system. To increase the statistical significance of our results, we use 6-fold cross-validation. Additionally, we carefully tuned the range of RRT-Connect to obtain the best results possible for all methods (range $= 2$ was used). A VICON camera system was used to gather object poses for SPARK. The Fetch's depth camera was used to build an octomap [8] for FLAME. Note that FLAME can retrieve relevant experiences even from partial/noisy octomaps which is often the case for real systems. Both demonstrate significant improvement even with few given examples.

VI. Discussion

We presented two experience-based planning frameworks for arbitrary manipulators in 3D environments, instantiated in SPARK and FLAME. SPARK uses an exact decomposition of the workspace into pairs of obstacles and achieves the best performance when this information is available. FLAME uses an approximate decomposition from a volumetric octree generated from sensor data and is comparable in performance with SPARK. We demonstrated that, with only few examples, both approaches confer significant improvement on planning time on a Fetch robot (8 DOF), on challenging manipulation tasks in environments that substantially vary, outperforming other learning-based approaches. One limitation of our methods is that due to the conservatism of the metrics, it is hard to generalize to environments that are far outside of the training distribution. For example, the "Large Shelf" and "Real Shelf" share no local primitives (due to the height of the shelves) and thus do not transfer from one to the other. To address these issues, in the future, we will investigate learning local primitive representations, metrics, and criticality tests.

REFERENCES

[1] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[2] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," *IEEE Int. Conf. Robot. Autom.*, vol. 3, pp. 4420–4426, 2003.

[3] D. Coleman, I. A. Sucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *IEEE Int. Conf. Robot. Autom.*, 2015, pp. 900–905.

[4] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 7087–7094.

[5] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using local experiences for global motion planning," in *IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 8606–8612.

[6] R. Kumar, A. Mandalika, S. Choudhury, and S. S. Srinivasa, "LEGO: Leveraging experience in roadmap generation for sampling-based planning," *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2019.

[7] G. Tang and K. Hauser, "Discontinuity-sensitive optimal control learning by mixture of experts," in *IEEE Int. Conf. Robot. Autom.*, 2019, pp. 7892–7898.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[10] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. of Computational Geometry and Applications*, vol. 9, no. 4n5, pp. 495–512, 1999.

[11] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[12] V. Boor, M. H. Overmars, and a. V. D. Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, May 1999, pp. 1018–1023.

[13] J.-M. Lien, S. L. Thomas, and N. M. Amato, "A general framework for sampling on the medial axis of the free space," in *IEEE Int. Conf. Robot. Autom.*, vol. 3, 2003, pp. 4439–4444.

[14] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Int. Wksp. on the Algorithmic Foundations of Robotics*. Springer, 1998.

[15] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *Algorithmic Foundations of Robotics VI*. Springer, pp. 361–376.

[16] L. Tapia, S. Thomas, B. Boyd, and N. M. Amato, "An unsupervised adaptive strategy for constructing probabilistic roadmaps," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2009, pp. 4037–4044.

[17] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics: Science and Syst.*, 2005, pp. 105–112.

[18] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-Free and C-Obstacle in arbitrary dimension," in *Springer Tracts in Advanced Robot.*, vol. 86, 2013, pp. 297–312.

[19] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid prm sampling with a cost-sensitive adaptive strategy," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2005, pp. 3874–3880.

[20] M. Bhardwaj, S. Choudhury, B. Boots, and S. Srinivasa, "Leveraging experience in lazy search," in *Robotics: Science and Syst.*, 2019.

[21] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3671–3678.

[22] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-Graphs: Bootstrapping planning with experience graphs," in *Robotics: Science and Syst.*, 2012.

[23] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," vol. 5, no. 2, pp. 2594–2601, 2020.

[24] P. Lehner and A. Albu-Schaeffer, "The repetition roadmap for repetitive constrained motion planning," *IEEE Robot. Autom. Letters*, vol. 3, no. 3, pp. 3884–3891, 2018.

[25] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, vol. 2, 2004, pp. 1618–1623.

[26] ——, "Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning," in *Int. Wksp. on the Algorithmic Foundations of Robotics*. Springer, 2008, pp. 35–51.

[27] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," *IEEE Int. Conf. Robot. Autom.*, pp. 3757–3762, 2008.

[28] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, 2001, pp. 1469–1474.

[29] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2018, pp. 3654–3661.

[30] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song, "Learning to plan in high dimensions via neural exploration-exploitation trees," in *Int. Conf. on Learn. Repr.*, 2020.

[31] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *IEEE Int. Conf. Robot. Autom.*, 2019, pp. 2118–2124.

[32] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," in *Robotics: Science and Syst.*, 2019.

[33] R. Terasawa, Y. Ariki, T. Narihira, T. Tsuboi, and K. Nagasaka, "3d-cnn based heuristic guided task-space planner for faster motion planning," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2020, pp. 9548–9554.

[34] K. Hauser and S. Emmons, "Global redundancy resolution via continuous pseudoinversion of the forward kinematic map," *IEEE Trans. on Autom. Science and Eng.*, vol. 15, no. 3, pp. 932–944, 2018.

[35] M. Farber, "Topological complexity of motion planning," *Discrete and Computational Geometry*, vol. 29, no. 2, pp. 211–221, 2003.

[36] W. Merkt, V. Ivan, T. Dinev, I. Havoutis, and S. Vijayakumar, "Memory clustering using persistent homology for multimodality-and discontinuity-sensitive learning of optimal control warm-starts," *arXiv preprint arXiv:2010.01024*, 2020.

[37] S. Finney, L. P. Kaelbling, and T. Lozano-Pérez, "Predicting partial paths from planning problem parameters." in *Robotics: Science and Syst.*, 2007.

[38] J.-M. Lien and Y. Lu, "Planning motion in environments with similar obstacles," *Robotics: Science and Syst.*, 2009.

[39] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *IEEE J. Robot. Autom.*, vol. 34, no. 2, pp. 111–127, 2013.

[40] D. Molina, K. Kumar, and S. Srivastava, "Learn and link: Learning critical regions for efficient planning," in *IEEE Int. Conf. Robot. Autom.*, 2020.

[41] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *IEEE Int. Conf. Robot. Autom.* IEEE, 2020, pp. 9535–9541.

[42] S. Brin, "Near neighbor search in large metric spaces," in *Int. Conf. on Very Large Data Bases*, 1995, pp. 574–584.

[43] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Autom.*, vol. 4, no. 2, pp. 193–203, 1988.

[44] J. J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *IEEE Int. Conf. Robot. Autom.*, vol. 4, 2004, pp. 3993–3998.

[45] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robot. Autom. Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[46] Z. Kingston and L. E. Kavraki, "Robowflex: Robot motion planning with *MoveIt* made easy," *IEEE Robotics and Automation Letters*, 2021, under Review. [Online]. Available: https://arxiv.org/abs/2103.12826

[47] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, 2000, pp. 995–1001.