2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Sampling Ensemble for Asymptotically Complete Motion Planning with Volume-Reducing Workspace Constraints

Sihui Li* Matthew A. Schack* Aakriti Upadhyay Neil T. Dantam

***Abstract*— Many robot tasks impose constraints on the workspace. For example, a robot may need to move a container without spilling its contents or open a door following the doorknob's arc. Such constraints may induce narrow volumes in the configuration space, traditionally a challenge for sampling-based methods, and further cause infeasibility. We extend sample-driven connectivity learning (SDCL), a robust approach for planning with narrow passages, to develop a sampling ensemble for workspace constraints. In particular, the ensemble combines SDCL, projection via dual quaternion optimization, and random sampling. These complementary sampling approaches support efficient and robust planning under workspace constraints. Further, this framework offers the ability to determine infeasibility under workspace constraints, which is unaddressed by previous constrained planning methods.**

I. Introduction

Many robot tasks require motion that not only avoids collision but also satisfies certain workspace constraints. For example, a robot moving a liquid-filled cup must keep it upright (see Figure 1), while a robot operating a door must follow paths constrained by the door's handle and hinges. Such constraints limit the valid robot poses and consequently limit the portion of the configuration space containing valid plans. Some prior approaches viewed constraints as manifolds in the configuration space [1], [2]; during planning, these approaches project configurations onto the manifold to produce valid plans. In this work, we consider volume-reducing constraints representing sub-regions of the free region of the configuration space. Some tasks inherently do not require strict manifold constraints. For example, holding a cup sufficiently upright to avoid spilling does not typically require perfect vertical alignment. Even tasks that seem to impose manifold constraints, such as opening a door or drawer, could be reframed as volume-reducing constraints based on flexibility in the arm through an adaptive control layer [3] or compliant hardware system [4]. Thus, volume-reducing constraints support a variety of useful robot tasks.

Volume-reducing constraint decrease the acceptable region in the configuration space, potentially creating more *narrow passages* and increasing planning difficulty due to small volumes and low sampling probabilities. To resolve this planning challenge, we adapt Sample-Driven Connectivity Learning (SDCL) [5], which learns connectivity in the configuration space to generate narrow passage samples, to now address volume-reducing constraints. The degenerate case of narrow passages occurs when obstacles and constraints create an infinitesimal (zero-volume) passage and consequent infeasibility of planning. To decide whether a plan exists or not with constraints, we need to construct infeasibility proofs. We address narrow passages and possible infeasibilities through an asymptotically complete planing approach, which returns a plan or an infeasibility proof in the limit [6].

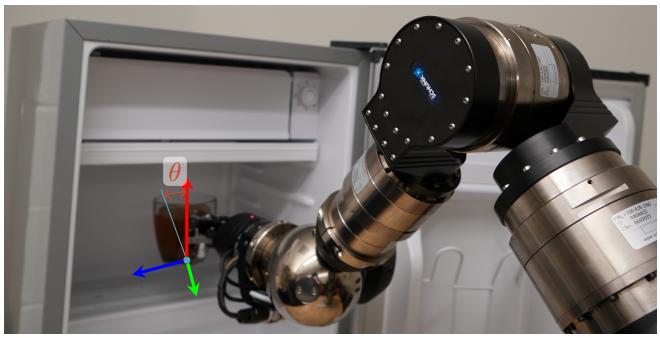


Fig. 1: A volume-reducing workspace constraint. The robot must transport the container while holding it sufficiently upright (within θ) to prevent spilling. Constrained motion within a confined area presents challenges for planning due to narrow passages and possible infeasibility.

We develop an asymptotically complete algorithm for motion planning with volume-reducing constraints and demonstrate the algorithm's effectiveness. First, we generally view volume reducing constraints as restrictions on the free space (see Sec. III), which supports use of blackbox validity checking typical of sampling-based planners [7], [8]. Second, we adapt the framework of SDCL (see Sec. IV), which supports efficient planning through narrow passages [5]. Third, we develop an approach to project sampled configurations into the constraint region (see Sec. V-B), which more efficiently satisfies constraints compared to uniform sampling. Fourth, we analyze this formulation and approach to show that it is asymptotically complete Sec. V-C. Finally, we evaluate this work in several robot manipulation scenarios (see Sec. VI).

A key step in our approach is the projection of configurations into the constraint region. We formulate this projection as a nonlinear program to find valid configurations. We begin from a sampled configuration that is invalid (constraint-violating) and then apply local, gradient-based methods to ensure the workspace constraint is satisfied.

The authors are with the Department of Computer Science, Colorado School of Mines, Golden, CO, USA. Email: {li,mschack,aakriti.upadhyay,ndantam}@mines.edu.
This work was supported in part by the ARL TBAM-CRP [W911NF-22-2-0235], ARL DCIST CRA [W911NF-17-2-0181, and Office of Naval Research grant N00014-21-1-2418.
* contributed equally.

This projection improves our ability to sample constraint-satisfying configurations compared to uniform sampling alone, especially for low-volume constraint regions.

We perform experiments in scenes using robots with 5-8 degrees of freedom (DOF) and workspace constraints representing holding a container upright, keeping the end-effector within a camera's field of view, and moving along a Cartesian plane while adhering to end-effector pose constraints. We compare our approach against baseline planners in OMPL [9], demonstrating improved time and robustness for difficult (narrow passage) scenes and the capability to determine infeasiblity when no plan exists.

II. Related Work

Sampling-based motion planners, such as probabilistic roadmaps (PRM) [7] and rapidly exploring random trees (RRT) [10], are effective and widely-used approaches. Generally, such planners sample robot configurations and grow a tree or graph towards the sample. However, challenges may arise when applying these methods to problems with small volumes (narrow passages) in the configuration space—which may be exacerbated by further constraints on the motion—due to low probability of sampling in, or connecting through, the small volumes. We address this challenge by developing a new sampler based on our SDCL algorithm [5], coupled with a projection approach, to generate samples in the low-volume regions.

A. Narrow Passages & Guided Sampling

When the configuration space contains narrow passages, uniform random sampling has low probability of generating the necessary configurations to find valid plans, so guided sampling approaches are often used. Bridge test sampling [11] increases the sampling density in narrow passages based on local information around the configurations. Some previous works used topological methods to guide sampling [12], [13] closer to objects or in narrow passages. KPIECE [14] uses multi-level grids in the search space to guide sampling in less explored areas. Visibility [15] and sparsity [16]-based algorithms guide sampling using the coverage information of the free space. Other approaches try to learn from previous samples to guide current sampling [17]–[21].

In this work, we adapt a guided sampling-based approach based on SDCL [5] to find plans through the narrow passages induced by volume-reducing constraints. Prior results showed that SDCL offers robust performance for planning through narrow passages in unconstrained manipulation problems, and the structures produced by SDCL effectively integrate with infeasibility proofs to offer asymptotic completeness. We now generalize SDCL as a planner-independent sampling strategy and couple it with projections to guide sampling into constraint regions.

B. Constrained Motion Planning

Constrained motion planning places certain limitations on a robot's motion. Constraints can model range of systems and are used for parallel robots [22], [23], grasping and manipulation [24], [25], computational biology and molecular simulations [26], [27], and animation [28], [29]. For example, we may specify constraints that a grasped container is held upright to avoid spilling or that an end-effector maintains contact with a surface such as for cleaning or painting.

Several forms of constraints are used for motion planning. In this work, we focus on volume-reducing constraints, represented as inequalities or intervals that limit valid volumes of the configuration space. Related to volume-reducing constraints are soft constraints [30]–[32], which also permit a volume of feasible values while favoring some particular value, e.g., filling a water pitcher under a faucet where closer to alignment to vertical allows the pitcher to hold more water. While our current work does not directly address soft constraints, such favored values could be incorporated into our projection approach. Other constraints may create lower dimension manifolds within the configuration space [1], [33], [34]. While some techniques in this current work could apply to such manifold constraints, analysis of infeasibility and asymptotically complete motion planning for lower dimension manifolds remains an area of future work.

Sampling-based methods must address constraint satisfaction for two operations: sampling configurations and connecting configurations; Kingston et. al. [2] classify five techniques to support constraints: relaxation [35]–[37], projection [1], [30], [33], [38], tangent-space sampling [33], [39], [40], incremental atlas construction [41]–[43], and reparameterization [44], [45]. In this work, we take a projection approach, which is similar to several previous works. Yao and Gupta applied projections to general end-effector constraints [46]. Task-constrained RRT [33], [39] and Constrained Bi-directional RRT (CBiRRT) [1], [47] address constraints via gradient descent based on the manipulator Jacobian pseudo-inverse. Similarly, Kunz and Stilman [30] address soft constraints via gradient descent projection. [38] constrains grasps, approach directions, and object transport paths by defining a planning margin based on grasp quality and success and finding solutions using Nelder-Mead and Jacobian pseudoinverse methods. Implicit manifold configuration spaces (IMACS) [34] define an implicit configuration space based on manifold constraints and incorporate projection onto the manifold. Our work now applies dual quaternion analysis [48] to formulate projection as a nonlinear program, supporting general workspace constraints, composition of multiple constraints, and the use of highly-engineered, robust, and efficient solution techniques [49]–[52].

C. Infeasibility

Some previous works construct exact path non-existence guarantees for single, rigid objects in a 2D or 3D workspace [53], [54]. Others [55], [56] use computational geometry tools to construct separations in the obstacle regions of the configuration space. Deterministic sampling-based motion planning also guarantees plan non-existence to some extent if no plans are found with the sampling coverage [57]–[60]. Previous works have also applied learning-based methods to

predict infeasible plans [61]–[63]. However, these methods do not provide definitive plan nonexistence guarantees.

In our previous work [64], [65], we proposed a sampling and learning-based infeasibility proof construction algorithm. The algorithm uses sampled configurations to learn a manifold in the configuration space and try to form a closure in the obstacle region to prove plan infeasibility. We apply this algorithm to motion planning problems with volume-reducing constraints to show infeasibility in some scenes.

III. Problem Definition

We address motion planning with volume reducing constraints. We first state the definition of unconstrained motion planning. Then, we incorporate volume reducing constraints.

An unconstrained motion planning problem [66] consists of a configuration space $\mathcal{C}$ of dimension n, a start configuration $\boldsymbol{q}_{\text{start}}$, and a goal configuration $\boldsymbol{q}_{\text{goal}}$. The configuration space $\mathcal{C}$ is the union of the disjoint obstacle region $\mathcal{C}_{\text{obs}}$ and free space $\mathcal{C}_{\text{free}}$. Both $\boldsymbol{q}_{\text{start}}$ and $\boldsymbol{q}_{\text{goal}}$ are in $\mathcal{C}_{\text{free}}$. Typically, sampling-based motion planners [9] implicitly define $\mathcal{C}_{\text{obs}}$ and free space $\mathcal{C}_{\text{free}}$ via blackbox collision checkers (such as [67]) that test whether a specific configuration is in $\mathcal{C}_{\text{obs}}$ or $\mathcal{C}_{\text{free}}$. The output of motion planning is plan σ through free space such that $\sigma[0,1] \in \mathcal{C}_{\text{free}}$, $\sigma[0] = \boldsymbol{q}_{\text{start}}$, $\sigma[1] = \boldsymbol{q}_{\text{goal}}$.

We consider volume reducing constraints of the form,

$$\mathcal{G}_i\left(\boldsymbol{q}\right) \leq \varepsilon_i,\ i = 1, \ldots, n\ , \tag{1}$$

where each $\mathcal{G}_i : \mathcal{C} \mapsto \mathbb{R}$ is a scalar function and ε_i defines the volume in which this constraint is satisfied. For example, the constraint to hold a container upright within tolerance ε would be $|\ln \hbar(\boldsymbol{q})| \leq \varepsilon$, where $\hbar$ is the rotation quaternion of the container relative to upright as a function of configuration $\boldsymbol{q}$ and which we determine from the robot's forward kinematics. Configurations satisfying constraints are $\mathcal{C}_{\text{in}} = \{\, \boldsymbol{q} \in \mathcal{C} \mid \mathcal{G}_i\left(\boldsymbol{q}\right) \leq \varepsilon_i, \forall i = 1, \ldots, n\}$. Configurations violating constraints are $\mathcal{C}_{\text{out}} = \mathcal{C} \setminus \mathcal{C}_{\text{in}}$. While we might consider constraints similarly to the implicitly defined obstacle region based on blackbox validity checking, it is useful to explicitly define constraints of the form in (1) to support the projections described in Sec. V-B.

A constrained motion planning problem consists of an unconstrained motion planning problem, coupled with a set of constraints $\mathcal{G}_1, \ldots, \mathcal{G}_n$ of the form in (1). To achieve asymptotic completeness, our probability of terminating with a plan or infeasibility proof must approach one in the limit [6]. That is, when a plan exists, the output is a plan σ such that $\sigma[0,1] \in \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$, $\sigma[0] = \boldsymbol{q}_{\text{start}}$, $\sigma[1] = \boldsymbol{q}_{\text{goal}}$. When there is no feasible plan, the output is an infeasibility proof $\mathcal{M}$ consisting of a closed manifold lying entirely in $\mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{out}}$ and separating $\boldsymbol{q}_{\text{start}}$ and $\boldsymbol{q}_{\text{goal}}$.

IV. Background

We briefly summarize key details of SDCL [5] and asymptotically complete motion planning [64], [65] on which this current work is based.

A. Sample-Driven Connectivity Learning (SDCL)

SDCL integrates sampling-based planning and machine learning to effectively produce samples in narrow passages [5]. There are two main steps in SDCL: learning a manifold and sampling the manifold. In the learning step, SDCL uses samples in the configuration space as training data for a classifier. All samples in the $\boldsymbol{q}_{\text{goal}}$ component are one class, and all other samples are the other class. The result of learning is a configuration space manifold $\mathcal{M}(\boldsymbol{q})$ ($\boldsymbol{q} \in \mathcal{C}$) that separates the configuration space into two parts. This learning process encodes in the manifold connectivity information from the samples of the configuration space. Next, SDCL samples points on the manifold. Sampled manifold points in $\mathcal{C}_{\text{free}}$ offer potential connections between the goal and non-goal components, effective to find plans through narrow passage.

B. Infeasibility Proof and Asymptotically Complete

An infeasibility proof is a closed manifold in $\mathcal{C}$ that is entirely in $\mathcal{C}_{\text{obs}}$ and that separates $\boldsymbol{q}_{\text{start}}$ and $\boldsymbol{q}_{\text{goal}}$ [64], [65]. Such a manifold shows that no path connecting $\boldsymbol{q}_{\text{start}}$ and $\boldsymbol{q}_{\text{goal}}$ is collision-free, so we can conclude with a path nonexistence guarantee. The infeasibility proof algorithm uses the same learned manifold $\mathcal{M}$ as SDCL and checks whether this manifold forms an infeasibility proof. Checking the manifold takes two steps. First, we triangulate the manifold into a piece-wise linear approximating polytope. Second, we check if each facet of the polytope is entirely in $\mathcal{C}_{\text{obs}}$.

Completeness is an important property for motion planners. A complete planner returns a plan or reports plans nonexistence in finite time. Many sampling-based motion planners are probabilistically complete [7], [8], meaning they find a plan in the limit for feasible cases. In infeasibility proof construction, as more points are sampled in $\mathcal{C}_{\text{free}}$, the learned manifold is guaranteed to converge to an infeasibility proof if no plan exists [6]. Combining a probabilistically complete sampling-based algorithm with infeasibility proof construction offers *asymptotic completeness* [6], meaning the planner finds either a plan or infeasibility proof in the limit.

V. Algorithm

In this work, we adapt SDCL and infeasibility proof construction to address volume-reducing workspace constraints. The result is an asymptotically complete motion planner supporting workspace constraints.

The key feature is a complementary ensemble of samplers to efficiently find valid points that satisfy constraints, i.e., in $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$. Uniform random sampling of configurations would have low probability of sampling narrow regions of $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$. Instead, we combine multiple sampling strategies to robustly find valid configurations. First, one thread samples narrow passage points similarly to SDCL (see Sec. V-A and Alg. 1). Second, one thread further projects points from $\mathcal{C}_{\text{out}}$ into $\mathcal{C}_{\text{in}}$ (see Sec. V-B and Alg. 2). Third, one thread performs random sampling (see Alg. 3). When the planner needs a new sample, we check in turn for samples produced by each of these threads (see Alg. 4). The SDCL

sampler robustly finds points through narrow passages. The projection sampler effectively finds constraint-satisfying points. The random sampler offers further space coverage over projections, supporting connectivity learning performed by SDCL. Together, this ensemble efficiently and robustly finds useful samples for constrained motion planning.

Additionally, planning under constraints requires valid (within $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$) tree/roadmap edges. Some approaches verify edges with the local planner [8]. Rather than modifying the local planner for every distinct motion planner, we instead adjust the validity checker to test whether a configuration is simultaneously in $\mathcal{C}_{\text{free}}$ and in $\mathcal{C}_{\text{in}}$. This modification further ensures that infeasibility proof construction checks for manifold containment in $\mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{out}}$.

A. SDCL Constraint Sampler

In this work, we adapt SDCL to both operate as a sampler and incorporate volume-reducing constraints. SDCL (see Alg. 1) operates with any multi-directional sampling-based motion planner such as a PRM [7]. In Alg. 1, P_{goal} are all graph nodes connectable to $\boldsymbol{q}_{\text{goal}}$, and P_{rest} are all the other graph nodes. We learn the manifold $\mathcal{M}$ from these two classes (line 4) and then sample the manifold. SDCL saves all sampled manifold points in $\mathcal{C}_{\text{free}}$ to a set B (line 7). In a separate thread, Alg. 2 projects points into the constraint region. When the sampler (see Alg. 4) is called from a planner, it checks first for valid points from SDCL's manifold and projected points; if no points exist, the sampler falls back to random sampling.

Algorithm 1: SDCL Sampling Thread

Input: $\boldsymbol{g}$, Q // Planning graph, All sampled config
Output: B_1 // Valid configurations for Alg. 4

1 **repeat**
2 $P_{\text{goal}} \leftarrow$ all nodes of $\boldsymbol{g}$ connectable to $\boldsymbol{q}_{\text{goal}}$
3 $P_{\text{rest}} \leftarrow \boldsymbol{g} \setminus P_{\text{goal}}$ // Other nodes
4 $\mathcal{M} \leftarrow \texttt{Learn}(P_{\text{rest}}, P_{\text{goal}})$
5 **foreach** $\boldsymbol{q} \in Q$ **do** // sample in parallel
6 $\mathbf{q}_{\text{m}} \leftarrow \texttt{SampleManifold}(\boldsymbol{q}, \mathcal{M})$
7 **if** $\mathbf{q}_{\text{m}} \in \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ **then** $B_1 \leftarrow B_1 \cup \{\mathbf{q}_{\text{m}}\}$
8 **until** `plan or infeasibility proof found`

Algorithm 2: Projection Sampling Thread

Input: $\mathcal{G}$ // Constraints
Output: B_2 // Valid configurations for Alg. 4

1 **repeat** // Solve Equation 2
2 $r \leftarrow \texttt{RandomSample}()$
3 $s \leftarrow \underset{\text{from } r}{\texttt{Solve}} \begin{pmatrix} \max_{\boldsymbol{q}} & 0 \\ \text{s.t.} & \mathcal{G}_i(\boldsymbol{q}) \leq \varepsilon_i \ \forall i \leq n \end{pmatrix}$
4 **if** $s \in \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ **then** $B_2 \leftarrow B_2 \cup \{s\}$
5 **until** `plan or infeasibility proof found`

Algorithm 3: Random Sampling Thread

Output: B_3 // Valid configurations for Alg. 4

1 **repeat**
2 $s \leftarrow \texttt{RandomSample}()$
3 **if** $s \in \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ **then** $B_3 \leftarrow B_3 \cup \{s\}$
4 **until** `plan or infeasibility proof found`

Algorithm 4: Sample

Input: B_1 B_2, B_3 // Valid configs from samplers
Output: s // A valid sample

1 **loop**
2 **if** $B_1 \neq \emptyset$ **then return** $\texttt{pop}(B_1)$
3 **else if** $B_2 \neq \emptyset$ **then return** $\texttt{pop}(B_2)$
4 **else if** $B_3 \neq \emptyset$ **then return** $\texttt{pop}(B_3)$
5 **else**
6 $s \leftarrow \texttt{RandomSample}()$
7 **if** $s \in \mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ **then return** s

B. Projection into Constraint Region

We project sampled points into $\mathcal{C}_{\text{in}}$ using a local, gradient-based method. Specifically, we create a nonlinear program (NLP) to find points in $\mathcal{C}_{\text{in}}$. The NLP formulation is independent of the particular optimization algorithm, though our experiments (see Sec. VI) use a sequential least squares quadratic programming (SLSQP) approach [49]–[51], which is a quasi-Newton method that approximates the Hessian and needs only the gradient. Though numerical methods to compute gradients are possible, many constraints (including all in Sec. VI) are expressible analytically as limitations on the workspace—e.g., holding a container upright limits the possible orientations of the end effector. Analytic gradients of such workspace constraints are computable via the chain rule and manipulator Jacobian (see Sec. V-B.2).

1) Optimization Formulation: We formulate an NLP to project an input configuration $\boldsymbol{q}$ into constraint region $\mathcal{C}_{\text{in}}$. We only need valid samples in $\mathcal{C}_{\text{in}}$, so we express the NLP using constraints $\mathcal{G}$ and a constant objective. As a result, we only apply the projection when the input configuration does not satisfy the constraints—i.e., $\boldsymbol{q} \notin \mathcal{C}_{\text{in}}$.

$$\begin{aligned} \max_{\boldsymbol{q}} \quad & 0 \\ \text{s.t. } & \mathcal{G}_i(\boldsymbol{q}) \leq \varepsilon_i \ \forall i \leq n \end{aligned} \tag{2}$$

We do require gradients to solve this NLP using SLSQP. While there is no guarantee that analytic gradients will exist for arbitrary $\mathcal{G}$, we can often effectively use numerical approximations (finite difference). Further, analytic gradients are computable for many workspace constraints.

2) Workspace Constraints: Many robot tasks impose workspace constraints that limit translation and/or orientation. We consider a special Euclidean group $\mathcal{SE}(3)$ workspace. Such a constraint $\mathcal{G}_i$ is then expressible as,

$$\mathcal{G}_i(\boldsymbol{q}) = \mathcal{F}_i\left({}^0\mathcal{S}_f(\boldsymbol{q})\right) , \tag{3}$$

where ${}^0\mathcal{S}_f(\boldsymbol{q}) \in \mathcal{SE}(3)$ represents the workspace pose of coordinate frame f, and $\mathcal{F}_i : \mathcal{SE}(3) \mapsto \mathbb{R}$ is a workspace constraint on frame f. The chain rule reduces the gradient of $\mathcal{G}_i$ to a function of workspace constraint gradient $\nabla\mathcal{F}_i$ and manipulator Jacobian $\mathbf{J}$.

$$\begin{aligned}\nabla\mathcal{G}_i(\boldsymbol{q}) &= \nabla\mathcal{F}_i\left({}^0\mathcal{S}_f(\boldsymbol{q})\right) * \frac{\partial}{\partial \boldsymbol{q}}{}^0\mathcal{S}_f(\boldsymbol{q}) \\ &= \nabla\mathcal{F}_i\left({}^0\mathcal{S}_f(\boldsymbol{q})\right) * \mathbf{J}(\boldsymbol{q}) \\ &= \left(\mathbf{J}(\boldsymbol{q})^T\left(\nabla\mathcal{F}_i\left({}^0\mathcal{S}_f(\boldsymbol{q})\right)\right)^T\right)^T\end{aligned} \tag{4}$$

Sec. VI-A describes several specific constraints $\mathcal{F}$.

We note that multiple representations of pose ${}^0\mathcal{S}_f$ and Jacobian $\mathbf{J}$ are possible, based generally on the structure of $\mathcal{SE}(3)$. However, one typical form of the manipulator Jacobian, relating workspace and configuration space *velocities* (i.e., $\begin{bmatrix}\omega & \dot{v}\end{bmatrix}^T = \mathbf{J}_{\mathbf{x}}\dot{\boldsymbol{q}}$), is not directly applicable to (4). Instead, we use dual number quaternions for poses because we can robustly take their Jacobians [48]. Dual quaternion pose $\mathcal{S}$ is,

$$\mathcal{S} = \hbar + \frac{1}{2}\vec{v} \otimes \hbar\boldsymbol{\varepsilon}\ , \tag{5}$$

where $\hbar$ is the rotation ordinary quaternion, $\vec{v}$ is the translation vector, and $\boldsymbol{\varepsilon}$ is the dual number element ($\boldsymbol{\varepsilon}^2 = 0$, $\boldsymbol{\varepsilon} \neq 0$). The manipulator Jacobian in dual quaternion form is then,

$$\mathbf{J} = \frac{\partial \mathcal{S}}{\partial \boldsymbol{q}} = \frac{1}{2}\left[\mathcal{S}\right]_R \mathbf{J}_\Omega \tag{6}$$

where $\left[\mathcal{S}\right]_R$ is the right matrix form of dual quaternion multiplication and $\mathbf{J}_\Omega$ is the twist form of the Jacobian (i.e., $\begin{bmatrix}\omega & \dot{v} + \vec{v} \times \omega\end{bmatrix}^T = \mathbf{J}_{\boldsymbol{\Omega}}\dot{\boldsymbol{q}}$) [68]. We refer the reader to [48] for further details on dual quaternions for robot kinematics.

C. Completeness Analysis

We discuss the requirements for asymptotic completeness of our algorithm, i.e., that in the limit we find a plan when one exists or prove infeasibility when no plan exists.

1) ε-goodness for $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{in}}$: Generally, probabilistic completeness of sampling-based planners requires the ε-goodness property [69] for $\mathcal{C}_{\text{free}}$ or similar notion of δ-clearance [8], [70], which means $\mathcal{C}_{\text{free}}$ has a "reasonably large" volume for the path to exist. ε-goodness and δ-clearance pose the same requirements but defined for different components of motion planning; in this paper, we use ε-goodness since we need to discuss a region of the configuration space directly.

We require ε-goodness for $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$, such that we have a volume to sample free space and constraint-satisfying configurations and form a path. In most cases, ε-goodness for $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ means $\mathcal{C}_{\text{in}}$ needs to be ε-good, though it is possible for infinitesimal regions of $\mathcal{C}_{\text{in}}$ to be subsumed within non-infinitesimal regions of $\mathcal{C}_{\text{free}}$, or vice-versa.

2) ε-blocked for $\mathcal{C}_{\text{obs}}$ and $\mathcal{C}_{\text{out}}$: Previously, we introduced the ε-blocked property for $\mathcal{C}_{\text{obs}}$ to guarantee that $\mathcal{C}_{\text{obs}}$ has sufficient "thickness" to support learning the infeasibility proof [6]. We require the same ε-blocked property for $\mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{out}}$, that is, the union of the region not satisfying the volume-reducing constraints and the obstacle region cannot be infinitesimal. For example, we do not consider cases where $\mathcal{C}_{\text{out}}$ is a set of disjoint points in $\mathcal{C}_{\text{free}}$, such as avoiding contact with a single hazardous point or avoiding kinematic singularities.

3) Asymptotic Completeness under volume-reducing constraints: With ε-good $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ and ε-blocked $\mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{out}}$, our algorithm is asymptotically complete. When a plan exists, the algorithm finds a plan in the limit, since $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ has a volume we can sample and the SDCL sampler incorporates random sampling if SDCL and projection are not effective. When a plan does not exist, samples generated in $\mathcal{C}_{\text{free}} \cap \mathcal{C}_{\text{in}}$ push the learned manifold into $\mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{out}}$ to form the infeasibility proof.

VI. Experiments

We evaluate our algorithm in scenes with various volume-reducing constraints and robots with 5 to 8 DOF (shown in Figure 2). The constraints differ in how much of the valid space they remove and whether they restrict the end effector's position (Sec. VI-A.1 and Sec. VI-A.2) or orientation (Sec. VI-A.3). We evaluate our results over 10 trials for each scene and compare the SCDL [5] sampler and SDCL with projection sampler find the narrow passages as well as with 7 baseline algorithms from OMPL [9], RRTConnect [8], PRM [7], LBKPIECE [14], EST [71], SBL [72], LBTRRT [73], BFMT [74].

We leverage parallelism in several parts of our algorithm and run our experiment on a multi-core system with NVIDIA TU102 GPU and a dual CPU AMD EPYC 7402 with 24 cores per CPU. We adapt PRM [75] in OMPL [9] to work with the SDCL thread. We solve the nonlinear optimization problems using sequential least-squares quadratic programming (SLSQP) [50], [51] in NLopt [76]. We train the RBF-kernel SVM using ThunderSVM [77], which supports GPU-accelerated SVM training. We check collisions using the Flexible Collision Library [67]. We model robot kinematics using Amino [78].

A. Test Scenes and Constraints

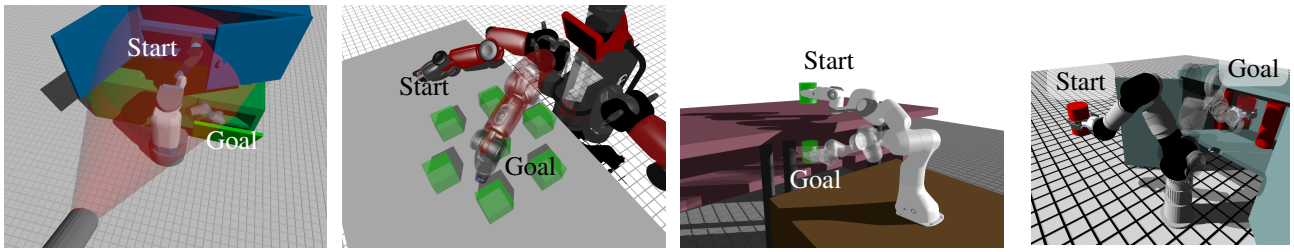
We evaluate our approach on four scenes with different volume-reducing constraints. Three scenes are feasible, and the final scene is infeasible.

*1) **Visibility Constraint**:* The scene in Figure 2a constrains the robot to keep its end effector within a camera's Field of View (FOV) in a kitchen environment.

We represent FOV as an infinitely extended cone with apex $\vec{v}_{\texttt{cone}} \in \mathbb{R}^3$, center axis represented as a unit vector $c_{\texttt{axis}} \in \mathbb{R}^3$, and viewing angle $c_\theta \in \left[0, \frac{\pi}{2}\right]$. We determine if a point lies within the cone's FOV by comparing the cosines of viewing angle c_θ and the angle between the cone's axis and the vector from the cone's origin to the input point $\vec{v}$,

$$\cos(c_\theta) \leq \mathcal{F}_{\texttt{cone}}(\vec{v}) \triangleq \frac{(\vec{v} - \vec{v}_{\texttt{cone}}) \cdot c_{\texttt{axis}}}{|(\vec{v} - \vec{v}_{\texttt{cone}})|}\ , \tag{7}$$

where $\vec{v}$ is the end effector's workspace position. Since $\mathcal{F}_{\texttt{cone}}$ is independent of orientation, its gradient is based solely the



(a) Kitchen Field of View (b) Planar Motion (c) Upright Shelf (d) Infeasible

Fig. 2: Three feasible test scenes and one infeasible scene with different robots and constraints. **(a)** Fetch must keep its end effector within the camera's Field of View. **(b)** Baxter must move its gripper at a constant height relative to the table. **(c)** Franka Panda must keep the cup upright while moving it between shelves. **(d)** An infeasible scene; Schunk LWA4D must move the cylinder into the cabinet without tilting beyond a limit.

end effector translation.

$$\nabla \mathcal{F}_{\texttt{cone}} = \frac{\partial}{\partial \vec{v}} \frac{(\vec{v} - \vec{v}_{\texttt{cone}}) \cdot c_{\texttt{axis}}}{|(\vec{v} - \vec{v}_{\texttt{cone}})|} \tag{8}$$
$$= \frac{c_{\texttt{axis}}}{|(\vec{v} - \vec{v}_{\texttt{cone}})|} - \frac{(\vec{v} - \vec{v}_{\texttt{cone}}) \cdot c_{\texttt{axis}}}{|(\vec{v} - \vec{v}_{\texttt{cone}})|^{3/2}} (\vec{v} - \vec{v}_{\texttt{cone}})$$

*2) **Planar Constraint**:* The scene in Figure 2b constrains a 7 DOF Baxter arm to be a fixed height above a table. We specify this constraint by requiring the distance between the end effector's height, z, and a reference height, $z_{\texttt{ref}}$ be less than some small number $\epsilon_{\texttt{plane}}$,

$$\epsilon_{\texttt{plane}} \geq \mathcal{F}_{\texttt{plane}}(z) \triangleq |z - z_{\texttt{ref}}|^2 \ . \tag{9}$$

The gradient of (9) via the chain rule is,

$$\nabla \mathcal{F}_{\texttt{plane}} = \frac{\partial}{\partial z} |z - z_{\texttt{ref}}|^2 = \frac{z - z_{\texttt{ref}}}{|z - z_{\texttt{ref}}|^2} \ . \tag{10}$$

*3) **Upright Constraint**:* The scene in Figure 2c constrains a 7 DOF Franka arm to hold a cup level while moving it between shelves. We specify this constraint based on relative rotation angle,

$$\epsilon_\theta \geq \mathcal{F}_{\texttt{up}}(\hbar) \triangleq |\ln(\hbar^* \otimes \hbar_{ref})|^2_{xy} \ , \tag{11}$$

where ϵ_θ specifies permissible rotation from vertical, $\hbar$ is the cup orientation quaternion, $\hbar_{ref}$ is the upright rotation quaternion, and $|(x, y, z)|^2_{xy} = x^2 + y^2$. The gradient of (11) follows from the methods described in [48].

*4) **Infeasible Scene**:* Figure 2d contains an infeasible scene to show that our algorithm generates infeasibility proofs when plans do not exist. The robot must move a cup from a position outside of the shelf to a position on the shelf. In this scene, we use the Schunk LWA4D, fixing the fifth and the last joints to limit to five DOF. Without upright constraints (11), a plan exists. With the upright constraint, no plan exists and the algorithm returns an infeasibility proof. We ran 20 trials, and all successfully proved infeasibility with a mean runtime of $60.40\,\text{s}$ and a standard deviation of $41.20\,\text{s}$.

B. Results

For the upright constraint and the planar constraint scenes, our algorithm is the fastest and most robust. We set the timeout to be 100 seconds; in most cases, the baseline motion planners cannot find a path within the time limit, which means the actual runtime difference to find paths is larger. These two scenes show improvement from our approach because the constraints creates narrow passages in the configuration space, showing that the projection and SDCL help resolve narrow passages well. In the FOV constraint scene, our algorithm is not the fastest, because the FOV constraint covers a whole region, as is shown in Figure 2a, which does not result in configuration space narrow passages.

Comparing the SDCL sampler with and without the projection, in the upright constraint scene, the projection improves the runtime by 44.9%. The other scenes do not show obvious differences when running SDCL with and without the projection. This means it is difficult to sample the upright constraint region with random sampling.

VII. Conclusion

We have presented an ensemble of samplers for robust planning under workspace constraints. This approach integrates sample-driven connectivity learning for robust planning through narrow passages, nonlinear programming to sample constraint-satisfying points, and random sampling to promote space coverage. In the tested scenes, baseline planners perform well for an easy scene (without narrow passages), while our approach offered time and robustness improvements for difficult scenes (containing narrow passages). Further, this algorithmic framework offers asymptotic completeness, meaning we can determine when motion planning is infeasible.

There are several possible avenues for future work. First, while the current work focused on volume-reducing hard constraints, soft constraints could be directly incorporated into the projection as an optimization objective within (2). Second, in our current implementation, infeasibility proofs are tractable for up to five DOF robots; in ongoing work, we are developing parallel, accelerated algorithms to scale infeasibility proofs to higher DOF. Finally, proving infeasibilities which are caused by lower dimension manifold constraints remains an area requiring further work and analysis.

Algorithm	Kitchen (Figure 2a)		Upright (Figure 2c)		Planar (Figure 2b)	
	Mean Runtime (s)	Completed	Means Runtime (s)	Completed	Mean (s)	Completed
ProjectSDCL	21.17 ± 23.59	10	**37.03 ± 16.79**	10	**12.88 ± 3.43**	10
SDCL	24.88 ± 21.12	10	67.23 ± 14.01	10	**11.38 ± 4.30**	10
PRM	34.32 ± 37.57	8	100.08 ± 0.04	0	100.07 ± 0.03	0
LBKPIECE1	3.33 ± 1.51	10	82.80 ± 23.23	4	100.04 ± 0.02	0
LBTRRT	100.14 ± 0.17	0	100.06 ± 0.03	0	100.04 ± 0.02	0
SBL	10.12 ± 3.86	10	100.07 ± 0.03	0	100.05 ± 0.02	0
BFMT	12.96 ± 2.11	10	100.40 ± 0.70	0	100.18 ± 0.18	0
EST	47.18 ± 30.95	9	100.07 ± 0.04	0	100.05 ± 0.03	0
RRTConnect	**0.78 ± 0.48**	10	100.07 ± 0.03	0	100.05 ± 0.03	0

TABLE I: The runtime and success rate of *ProjectSDCL* and the baseline methods for the kitchen (Figure 2a), upright (Figure 2c), and planar (Figure 2b) environments. The planar and upright environments were more constrained than the kitchen environment, making them more challenging to solve by purely sampling the space. Our work projects sampled points into the constraint regions, allowing us to more efficiently sample the space leading to faster performance on the more constrained scenes.

REFERENCES

[1] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *ICRA*, 2009, pp. 625–632.

[2] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018.

[3] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki, "Sampling-based motion planning for uncertain high-dimensional systems via adaptive control," in *WAFR*. Springer, 2021, pp. 159–175.

[4] M. Bonilla, E. Farnioli, L. Pallottino, and A. Bicchi, "Sample-based motion planning for robot manipulators with closed kinematic chains," in *ICRA*. IEEE, 2015, pp. 2522–2527.

[5] S. Li and N. T. Dantam, "Sample-Driven Connectivity Learning for Motion Planning in Narrow Passages," in *ICRA*. IEEE, 2023, pp. 5681–5687.

[6] ——, "Exponential convergence of infeasibility proofs for kinematic motion planning," in *WAFR*, 2022.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *T-RO*, vol. 12, no. 4, pp. 566–580, 1996.

[8] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000, pp. 995–1001.

[9] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *RAM*, vol. 19, no. 4, pp. 72–82, 2012.

[10] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, 2001.

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

[12] A. Upadhyay, B. Goldfarb, W. Wang, and C. Ekenna, "A new application of discrete morse theory to optimizing safe motion planning paths," in *WAFR*. Springer, 2023, pp. 18–35.

[13] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian, "Efficient path planning in narrow passages for robots with ellipsoidal components," *T-RO*, 2022.

[14] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *WAFR*. Springer, 2009, pp. 449–464.

[15] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[16] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *IJRR*, vol. 33, no. 1, pp. 18–47, 2014.

[17] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *ICRA*. IEEE, 2007, pp. 3307–3312.

[18] ——, "Sampling-based motion planning using predictive models," in *ICRA*. IEEE, 2005, pp. 3120–3125.

[19] S. Dalibard and J.-P. Laumond, "Linear dimensionality reduction in random motion planning," *IJRR*, vol. 30, no. 12, pp. 1461–1476, 2011.

[20] W. Wang, L. Zuo, and X. Xu, "A learning-based multi-RRT approach for robot path planning in narrow passages," *Journal of Intelligent & Robotic Systems*, vol. 90, no. 1, pp. 81–100, 2018.

[21] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *ICRA*. IEEE, 2018, pp. 7087–7094.

[22] H.-J. Su and J. M. McCarthy, "Dimensioning a constrained parallel robot to reach a set of task positions," in *ICRA*. IEEE, 2005, pp. 4026–4030.

[23] N. Zhang and W. Shang, "Dynamic trajectory planning of a 3-dof under-constrained cable-driven parallel robot," *Mechanism and Machine Theory*, vol. 98, pp. 21–35, 2016.

[24] R. Holladay, T. Lozano-Pérez, and A. Rodriguez, "Force-and-motion constrained planning for tool use," in *IROS*. IEEE, 2019, pp. 7409–7416.

[25] A. M. Zanchettin and P. Rocco, "Motion planning for robotic manipulators using robust constrained control," *Control Engineering Practice*, vol. 59, pp. 127–136, 2017.

[26] L.-Q. Yang, P. Sang, Y. Tao, Y.-X. Fu, K.-Q. Zhang, Y.-H. Xie, and S.-Q. Liu, "Protein dynamics and motions in relation to their functions: several case studies and the underlying mechanisms," *Journal of Biomolecular Structure and Dynamics*, vol. 32, no. 3, pp. 372–393, 2014.

[27] A. Upadhyay, T. Tran, and C. Ekenna, "A topology approach towards modeling activities and properties on a biomolecular surface," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2021, pp. 157–162.

[28] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motions for interactive object manipulation and grasping," in *ACM SIGGRAPH 2008 classes*, 2008, pp. 1–11.

[29] C. Schulz, C. von Tycowicz, H.-P. Seidel, and K. Hildebrandt, "Animating deformable objects using sparse spacetime constraints," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–10, 2014.

[30] T. Kunz and M. Stilman, "Manipulation planning with soft task constraints," in *IROS*, 2012, pp. 1937–1942.

[31] N. M. Ceriani, A. M. Zanchettin, P. Rocco, A. Stolt, and A. Robertsson, "Reactive task adaptation based on hierarchical constraints classification for safe industrial robots," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2935–2949, 2015.

[32] J. Wang, S. Liu, B. Zhang, and C. Yu, "Manipulation planning with soft constraints by randomized exploration of the composite configuration space," *International Journal of Control, Automation and Systems*, vol. 19, no. 3, pp. 1340–1351, 2021.

[33] M. Stilman, "Task constrained motion planning in robot joint space," in *IROS*. IEEE, 2007, pp. 3074–3081.

[34] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *IJRR*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[35] M. Bonilla, L. Pallottino, and A. Bicchi, "Noninteracting constrained motion planning and control for robot manipulators," in *ICRA*. IEEE, 2017, pp. 4038–4043.

[36] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato, "Resampl: A region-sensitive adaptive motion planner," in *WAFR*. Springer, 2008, pp. 285–300.

[37] J. Bialkowski, M. Otte, and E. Frazzoli, "Free-configuration biased sampling for motion planning," in *IROS*. IEEE, 2013, pp. 1272–1279.

[38] J. Huh, B. Lee, and D. D. Lee, "Constrained sampling-based planning for grasping and manipulation," in *ICRA*. IEEE, 2018, pp. 223–230.

[39] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *T-RO*, vol. 26, no. 3, pp. 576–584, 2010.

[40] M. Cefalo, G. Oriolo, and M. Vendittelli, "Task-constrained motion planning with moving obstacles," in *IROS*. IEEE, 2013, pp. 5758–5763.

[41] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle rrt: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.

[42] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *T-RO2013*, vol. 29, no. 1, pp. 105–117, 2013.

[43] L. Jaillet and J. Porta, "Asymptotically-optimal path planning on manifolds," in *RSS*, Sydney, Australia, July 2012.

[44] T. McMahon, S. Thomas, and N. M. Amato, "Sampling-based motion planning with reachable volumes for high-degree-of-freedom manipulators," *IJRR*, vol. 37, no. 7, pp. 779–817, 2018.

[45] L. Han, L. Rudolph, J. Blumenthal, and I. Valodzin, "Convexly stratified deformation spaces and efficient path planning for planar closed chains with revolute joints," *IJRR*, vol. 27, no. 11-12, pp. 1189–1212, 2008.

[46] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 316–327, 2007.

[47] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *IJRR*, vol. 30, no. 12, pp. 1435–1460, 2011.

[48] N. T. Dantam, "Robust and efficient forward, differential, and inverse kinematics using dual quaternions online," in *IJRR*. Springer, 2020.

[49] S. G. Johnson, "The NLopt nonlinear-optimization package," 2024, http://github.com/stevengj/nlopt.

[50] D. Kraft, "A software package for sequential quadratic programming," Institut für Dynamik der Flugsysteme, Oberpfaffenhofen, Tech. Rep. DFVLR-FB 88-28, July 1988.

[51] ——, "Algorithm 733: TOMP–fortran modules for optimal control calculations," *Transactions on Mathematical Software (TOMS)*, vol. 20, no. 3, pp. 262–281, 1994.

[52] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 10 2023. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[53] A. Varava, J. F. Carvalho, F. T. Pokorny, and D. Kragic, "Caging and path non-existence: a deterministic sampling-based verification algorithm," in *IJRR*. Springer, 2020, pp. 589–604.

[54] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," in *ICRA*, 2001.

[55] Z. McCarthy, T. Bretl, and S. Hutchinson, "Proving path non-existence using sampling and alpha shapes," in *ICRA*. IEEE, 2012, pp. 2563–2569.

[56] S. Li and N. T. Dantam, "Towards general infeasibility proofs in motion planning," in *IROS*, 2020, pp. 6704–6710.

[57] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," in *ICRA*, vol. 2. IEEE, 2001, pp. 1481–1487.

[58] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *IJRR*, vol. 37, no. 1, pp. 46–61, 2018.

[59] M. Tsao, K. Solovey, and M. Pavone, "Sample complexity of probabilistic roadmaps via ε-nets," in *ICRA*. IEEE, 2020, pp. 2196–2202.

[60] D. Dayan, K. Solovey, M. Pavone, and D. Halperin, "Near-optimal multi-robot motion planning with finite sampling," *T-RO*, 2023.

[61] A. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *RAM*, vol. 4, no. 2, pp. 1255–1262, 2019.

[62] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *ICRA*, 2020, pp. 9563–9569.

[63] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, "Learning geometric reasoning and control for long-horizon tasks from visual input," in *ICRA*, 2021, pp. 14 298–14 305.

[64] S. Li and N. T. Dantam, "A Sampling and Learning Framework to Prove Motion Planning Infeasibility," *IJRR*, 2023.

[65] ——, "Scaling Infeasibility Proofs via Concurrent, Codimension-one, Locally-updated Coxeter Triangulation," *RA-L*, 2023.

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

[67] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *ICRA*, 2012, pp. 3859–3866.

[68] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[69] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, "Randomized query processing in robot path planning," *JCSS*, vol. 57, no. 1, pp. 50–60, 1998.

[70] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, vol. 30, no. 7, pp. 846–894, 2011.

[71] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *ICRA*, vol. 3. IEEE, 1997, pp. 2719–2726.

[72] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics research*. Springer, 2003, pp. 403–417.

[73] O. Salzman and D. Halperin, "Asymptotically near-optimal rrt for fast, high-quality motion planning," *T-RO*, vol. 32, no. 3, pp. 473–483, 2016.

[74] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *IROS*. IEEE, 2015, pp. 2072–2078.

[75] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," vol. 12, no. 4, pp. 566–580, August 1996.

[76] S. G. Johnson and J. Schueller, "Nlopt: Nonlinear optimization library," *Astrophysics Source Code Library*, pp. ascl–2111, 2021.

[77] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, "ThunderSVM: A fast SVM library on GPUs and CPUs," *Journal of Machine Learning Research*, vol. 19, pp. 797–801, 2018.

[78] N. T. Dantam, "Robust and efficient forward, differential, and inverse kinematics using dual quaternions," *IJRR*, 2020.